Aus der Klinik und Poliklinik für Psychiatrie und
Psychotherapie der Universität Würzburg
Direktor: Professor Dr. med. Dr. h.c. Helmut Beckmann

Der Einfluss des Schweregrades eines neuropsychologischen Tests (Span-of-Apprehension-Test) auf kognitiv-evozierte Potentiale

Inaugural - Dissertation
zur Erlangung der Doktorwürde der
Medizinischen Fakultät
der
Bayerischen Julius-Maximilians-Universität zu Würzburg

vorgelegt von
Sven Dannemann
aus München
Würzburg, Mai 2002

Referent: Prof. Dr. med. Dr. h.c. H. Beckmann

Koreferent: Priv.-Doz. Dr. med. A. J. Fallgatter

Dekan: Prof. Dr. med. V. ter Meulen

Tag der mündlichen Prüfung: 28.01.2003

Der Promovend ist Arzt.

meinen Eltern

„Die Zeit aber, so lehrt der Blick auf die Sonne, ist selbst nur eine Darstellung des Lichts, in dem alles sich zeigt: Das Vergangene ist ebenso präsent wie das Zukünftige und das Gegenwärtige; In Vergangenheit, Zukunft und Gegenwart spielt eine Präsenz und eine Zugänglichkeit, die nichts anderes ist als die Möglichkeit, die Offenheit des Verstehens selbst.“

(Günter Figal)

Inhaltsverzeichnis

1 Einleitung

In der psychologischen Diagnostik, einem Teilgebiet der Psychologie, gelten psychometrische Tests, projektive Verfahren, psychophysiologische Methoden, die Ausdrucksdiagnostik und die Interaktionsdiagnostik als wichtigste Methoden.

"Ein psychologischer Test ist eine Art kleines Experiment, bei dem unter bestimmten Bedingungen Probanden oder Versuchspersonen [..] bestimmte Testaufgaben, sogenannte Testitems, ausführen; ihr Verhalten bzw. ihre Ergebnisse werden anhand vorgegebener Kriterien gemessen bzw. gewertet." [33]

Psychometrische Tests werden in Persönlichkeits- und Leistungstests eingeteilt, wobei man bei den letzteren zwischen Geschwindigkeits- und Niveautests unterscheidet.

Der Span-of-Apprehension-Test (SAT) untersucht sowohl Geschwindigkeit als auch Niveau.

Kognition bezeichnet die zwischen Wahrnehmung und Motorik ablaufenden Prozesse; die Auswirkungen solcher Prozesse auf das Auffassungsvermögen (engl.: span of apprehension) optischer Information ermittelt der SAT.

Der Span-of-Apprehension-Test ist also ein psychologischer Test, der zur Untersuchung der kognitiven visuellen Leistungsfähigkeit eines Individuums verwendet wird.

Das Original wurde von W. K. Estes und H. A. Taylor 1965 ausgearbeitet [10]und 1969 von J. M. Neale modifiziert [30]. Für diese Studie wurde eine computergesteuerte Version verwendet (SAT.C, Seite 54), die nach der 1978 von R. F. Assarnow und D. J. MacCrimmon beschriebenen Fassung programmiert wurde [1].

Die Kognition wird hier gemessen an der Anzahl richtiger Antworten und der durchschnittlich bis zu einer Antwort benötigten Zeit. Der SAT wurde auch in psychiatrischen Studien eingesetzt. So stellen Neale und Mc Intyre bereits 1969 dar, dass bei an Schizophrenie Erkrankten das Auffassungsvermögen nur die Hälfte Gesunder beträgt [30]. 1974 beschreibt Davidson die beim SAT erreichte Leistung von Schizophrenen als signifikant geringer [9]. In 1981 und 1982 veröffentlichten Studien stellen Asarnow und MacCrimmon fest, dass der SAT sensitiv für bestimmte schizophrene Dysfunktionen ist [2;3]. Diese Ergebnisse werden 1983 von Asarnow [4],

1985 von Harris [18], 1986 von Tarnowski [43], 1992 von Maier [26], 1993 von Hain [17] und 1997 von Voruganti [44], Ito [19] und Buchanan [7] bestätigt. Der SAT ist jedoch kein spezifischer Indikator für die Schizophrenie an sich wie Strauss 1987 [39] und Rund 1992 [32] aufzeigen.

Eine Verstärkung der diagnostischen Schärfe des SAT könnte möglicherweise durch die gleichzeitige Erfassung ereignis-korrelierter Potentiale (EKP) erreicht werden. Anlass zu dieser Annahme geben die bereits erfolgreich durchgeführten funktionellen EEG-Untersuchungen von z.B. Strik [41], Fallgatter [11-13] und Müller [29]. Strik formuliert 1993, unter welchen Umständen EEG-Epochen einen Aussagewert erhalten: “Kürzere Epochen sind der Untersuchung durch evozierte Potentiale (EKP) zugänglich, die jedoch nur durch Mittelung vieler EEG-Epochen erhalten werden. Dies impliziert 1) die Kenntnis eines Zeitpunktes, der in exaktem zeitlichen Verhältnis zu der untersuchten kognitiven Operation steht. Spontane Kognition kann durch diese Methode nur untersucht werden, wenn sie mit messbarem motorischen Verhalten einhergeht. 2) Die häufige Wiederholung einer Aufgabe, um eine ausreichendes Verhältnis zwischen Signal und Rauschen (signal-to-noise ratio) zu erhalten. Dadurch kann das Ergebnis jedoch von Zeit-Faktoren wie Habituation beeinflusst werden." [40]

1984 wird von Strandburg eine neurophysiologische Studie erstmals mit dem SAT durchgeführt, in der elektrisch evozierte Potentiale aufgezeichnet werden [34]. Weitere Studien, die EKP untersuchen, befassen sich vorwiegend mit dem Messparameter "endogene negative Aktivität" - 1990 Caplan [8] und 1991, 1993 und 1994 Strandburg [35-38].

Mit der vorliegenden Studie sollen weitere von Lehmann in den achtziger Jahren eingeführte EKP-Messparameter [22-25], die auch in oben genannten Studien [11-13;29;41] bestimmt werden, beim SAT Anwendung finden. Bei den Parametern handelt es sich um Deskriptoren, die zur quantitativen Analyse von Kartenlandschaften, welche hirnelektrische Felder repräsentieren, dienen. In der vorliegenden Studie werden die Methoden nach Lehmann und Strik[42] verwendet, da sie eine Analyse der EKP unabhängig von fixierten Elektrodenpositionen erlauben.

Die neuropsychologische Auswertung des SAT sollte bei korrekter Methodik eine verlängerte Reaktionszeit und eine verminderte Anzahl von richtigen Antworten bei höheren Schwierigkeitsgraden im Test ergeben.

Die Auswertung der während des SAT aufgezeichneten EKP führt hypothetisch bei höherem Schwierigkeitsgrad zu neurophysiologisch für gesteigerte Aufmerksamkeit typischen Potentialveränderungen. Es werden besonders Veränderungen der p150 und von späteren Potentialen (so P300) erwartet.

Bei Bestätigung dieser Hypothesen könnte ein SAT in Verbindung mit EKP Aufmerksamkeitsdefizite nachweisen. Klinische Anwendung könnte der SAT mit EKP-Aufzeichnung dann bei der Untersuchung von psychiatrischen Patienten mit beispielsweise schizophrenen Psychosen finden, die an allgemeinen Störungen der Aufmerksamkeit oder Auffassung leiden. Außerdem könnte er bei psychiatrischen Verlaufsuntersuchungen zur Erfassung des Krankheitsverlaufs herangezogen werden.

Andere Studien [35;36;38], die den SAT mit EKP-Aufzeichnung verwenden, stellen besonders seine Relevanz bei der Schizophreniediagnostik in den Vordergrund.

2 Methodik

2.1 Probanden

Als Versuchspersonen wurden neun Rechtshänder, bei denen keine psychiatrischen oder neurologischen Erkrankungen bekannt waren, ausgewählt. Zur Untersuchung der Händigkeit wurde der Edingburgh-Handedness-Task eingesetzt. Das mittlere Alter der fünf Männer und vier Frauen war 25,2 ± 1,4 Jahre (Spanne 23 - 28). Sie waren Probanden der Universität Würzburg und erklärten nach allgemeiner Aufklärung über Art und Gegenstand der Untersuchung freiwillig ihr Einverständnis. Keiner der Probanden stand unter psychotroper Medikation (einschließlich Koffein). Der Visus musste auf beiden Augen einwandfrei, nötigenfalls korrigiert sein. Um Interaktionen mit den Testergebnissen zu vermeiden, wurde über die genaue Aufgabe während der Untersuchungen erst unmittelbar vor Beginn aufgeklärt.

2.2 Stimulation und Paradigma

2.2.1 Kognitive Tests

Im Rahmen einer Studie wurden vier visuelle Informationsverarbeitungstests in einer Untersuchungssitzung durchgeführt: Ein Backward-Masking-Test (BMT), zwei verschiedene Versionen eines Span-of-Apprehension-Tests (SAT) und ein Digit-Span-Reverse-Matching-Test (DSRMT). Dieser Vorgang dauerte 75 Minuten. Jedem Test ging eine vom Versuchsleiter vorgelesene Erklärung und ein kurzer Probelauf voraus. Es folgte jeweils eine kurze Pause. Verständnisfragen wurden hier noch beantwortet. Während der Tests wurde keine Hilfestellung und keine Bestätigung gegeben. Die Buchstaben- und Zahlenfolgen, die in den Tests verwendet wurden, waren für alle Probanden dieselben und vor Studienbeginn durch spezielle Programme zufällig ausgewählt worden.

2.2.2 Span-of-Apprehension-Test

Für diese Studie wurde die von R. F. Assarnow und D. J. MacCrimmon 1978 beschriebene Version vom Verfasser 1997 als PC-Version “SAT” programmiert (Quellcode auf Seiten 54ff.).
Auf dem Monitor erschien eine 4 x 4 Matrix der Seitenlänge 18 cm. Diese blieb während des Tests unverändert bestehen. In jedem der 16 Felder der Matrix konnte ein Buchstabe zentriert abgebildet werden. In einem der Felder erschien immer der Buchstabe F oder der Buchstabe T. Diese werden im Folgenden auch Target-Buchstaben, Targets oder Trigger genannt. Gleichzeitig wurden in den übrigen Feldern zur Ablenkung andere Buchstaben gezeigt, die als Distraktor-Buchstaben oder Distraktoren bezeichnet werden. Distraktoren konnten alle übrigen Buchstaben des Alphabets sein, wobei nie zwei gleiche in einer Matrix auftauchten. Der Proband sollte die linke Maustaste drücken, wenn er ein F sah, die rechte, wenn er ein T sah.

Folgender Ablauf wird als „Trial“ bezeichnet:

1. Die Matrix erscheint mit Target und Distraktoren auf dem Monitor.
2. Die Antwort des Probanden erfolgt. Die Präsentationszeit betrug 100 ms.

Die Anzahl der Distraktoren war variabel. Es wurden null, zwei, vier oder neun Distraktoren gezeigt. Mit jeder Distraktorenanzahl wurden 50 Trials (25 mit Target F und 25 mit Target T) in zufälliger Reihenfolge aufgeführt. Die Reihenfolge war für jeden Probanden gleich und vor Studienbeginn durch das Computerprogramm “Trials” (Dannemann, Quellcode auf Seiten 66ff.) festgelegt. Ein Beispielausdruck der verwendeten Trials befindet sich in Abschnitt 8.4 auf Seite 69. Die Zeit zwischen zwei Trials betrug zwei Sekunden. Der Test beinhaltete also insgesamt 200 Trials und hatte zuzüglich der individuellen Reaktionszeiten eine Gesamtlänge von ca. 11 Minuten.

Den Probanden wurde dieser Erklärungstext vorgelesen:

“Im nächsten Test wird in einem Gitter ein F oder ein T gezeigt. Gleichzeitig sind andere Buchstaben zu sehen, unter denen sich aber **immer** ein F oder ein T befindet. Wenn ein F gezeigt wird, drücken Sie bitte die linke Maustaste. Wenn hingegen ein T gezeigt wird, drücken Sie bitte die rechte Maustaste. Es werden also beide Maustasten benötigt, bei F die linke, bei T die rechte.”

2.3 Testdurchführung

2.3.1 Ableitebedingungen

Die Tests wurden in einem abgedunkelten, klimatisierten, elektrisch abgeschirmten Raum bei maximal-möglicher Geräuschreduktion durchgeführt. Die Probanden saßen in 1,20 m Abstand vor einem entspiegelten 15-Zoll Monitor (Firma ADI), auf dem die Testaufgaben gezeigt wurden. So ergab sich ein Blickwinkel von 12.4 × 8.8 °. Die Buchstaben und Zahlen waren schwarz, von der Größe 1,3 cm und der Schriftart Sans Serif auf weißem Hintergrund abgebildet. Die Tests liefen auf einem Personalcomputer (Prozessor 80486) und waren zuvor speziell für diese Studie vom Verfasser der Abhandlung in der Programmiersprache C erstellt worden (siehe Abschnitt 7.1.1, Seite 52). Die Probanden beantworteten die Aufgabenstellungen mit einer Computermaus, die vor ihnen auf einem Tisch lag. Sie hatten die Anweisung, nur zu antworten, wenn sie sich sicher waren. Die Probanden waren durch eine 1,5 m hohe Wand vom Versuchsleiter getrennt, so dass sie diesen und die Rechner nicht sehen, aber seine Stimme bei eventuell notwendigen Anweisungen hören konnten.

2.3.2 EKP Aufzeichnung

Das EEG wurde mit 61 Elektroden (gold-cup, 5mm Durchmesser, befestigt mit TECA Elektrodenpaste) entsprechend dem Internationalen 10/10-System aufgezeichnet. Der Augenartefakterkennung dienten drei zusätzliche Elektroden, die an den äußeren Augenwinkeln und über dem rechten Jochbein angebracht wurden. Jedem Stimulus

wurde ein identifizierbarer Marker zugeordnet. Diese wurden in einem separaten Marker-Kanal während der EEG-Aufzeichnung erfasst. Der Aufnahme diente ein 80-Kanal DC-Verstärker (MI-System) und ein Daten-Aquisitions-Programm (Easys2). Die Kalibration erfolgte mit einem internen digitalen 100 µV / 10 Hz Sinus Signal. Der Hochfrequenzfilter wurde auf 70 Hz eingestellt, der Tieffrequenzfilter auf 0,15 Hz. Die Umwandlungsrate (A/D - Frequenz) betrug 250 Hz. Als Aufzeichnungs-Referenz diente der Durchschnittswert aller Elektroden (Gleichung 1: Average Reference, Seite 52). Die Impedanzen aller Elektroden waren unter 10 kΩ.

2.4 Datenanalyse

Die Daten wurden Off-Line, d.h. nach Abschluss der EEG-Aufzeichnung, auf Artefakte geprüft und in Epochen von je einer Sekunde eingeteilt. Epochen, die in einem oder mehreren Aufnahmekanälen (die bipolaren Augenkanäle eingeschlossen) 98 µV überschritten, wurden von der Weiterverarbeitung ausgeschlossen. Diese Aufgaben übernahm das Programm „Eamarkex“ (Strik et Dannemann, Quellcode auf Seiten 72ff.). Probanden, die weniger als 50% gültige Datenepochen unter einer der Bedingungen hatten, wurden von der Studie ausgeschlossen (n = 0).

Die Daten wurden anhand der Faktoren Stimulus (F / T) und Distraktorenanzahl (0 / 2 / 4 / 9) kategorisiert und separat zu ereigniskorrelierten Potentialen (EKP) gemittelt. Diese Berechnungen wurden mit den Computerprogrammen „Average“, „Epeak“ (Strik et Dannemann, Quellcodes auf Seiten 87ff. und 90ff.) und Eaintpol (Müller et Dannemann, Quellcode auf Seiten 78ff.) durchgeführt.

Die topographische Darstellung wird nach Lehmann durch räumliche Interpolation von simultanen Mehrkanalmessungen eines Messzeitpunktes und Verwendung von Isopotentiallinien aufgebaut [21]. Als Maß zur referenzunabhängigen Beschreibung aller Elektrodenpotentiale wurde die Globale Feldstärke (GFP, Abkürzung für Global Field Power) berechnet; diese entspricht der mittleren Standardabweichung aller Amplitudenwerte der Elektroden (Gleichung 2, Seite 52).[23] Mit den Minima der Grand Average GFP-Kurve über Stimulus und Distraktorenanzahl in der Spanne von 0 -

600 msec (Abbildung 3, Seite 11) wurden vier Zeitfenster (Tabelle 5, Seite 12) bestimmt. Für jedes Zeitfenster wurden GFP, Latenz, Centroidpositionen, Centroidenschwerpunkt, -distanz und -winkel berechnet:
Die Latenzen wurden definiert als die Zeitpunkte maximaler GFP innerhalb der Fenster. Für die Zeitpunkte der maximalen GFP im Bereich der genannten Zeitfenster wurde dann die Position der positiven und negativen Centroide des Feldes in anterior-posteriorer und links-rechts Richtung (4 Werte) berechnet. Die Centroide sind die nach Amplitude gewichteten Orte der positiven und negativen Kartenfläche nach Abzug des räumlichen DC-Offsets (average reference) auf einer planaren Projektion der Elektrodenpositionen (siehe hierzu auch Gleichung 3 auf Seite 52)[23]. Weiter wurde die Distanz als Länge der direkten Strecke zwischen dem positiven und negativen Centroid und der Winkel zwischen dieser Strecke und der Vertikalen ermittelt.

2.5 Statistik

Konfirmatorisch wurde eine 4x4x2 MANOVA (Multivarianzanalyse) für wiederholte Messungen durchgeführt. Dabei wurden die Faktoren Zeitsegment (vier Epochen), Distraktorenanzahl (null, zwei, vier oder neun) und Stimulus (F oder T) auf die vier Koordinaten, Distanz und Winkel der Centroidpositionen zur Untersuchung topografischer Effekte, der EKP-Amplituden (GFP) und der EKP-Latenzen angewandt. Post-hoc Tests wurden mit gepaarten zweiseitigen t-Tests und einem Signifikanzniveau von 5% durchgeführt.
Zur Berechnung der MANOVA wurde das Statistica© -Programm-Paket verwendet. Die Post-hoc Tests wurden mit Lstat (Strik und Müller) durchgeführt und in Excel© (Microsoft®) weiterverarbeitet.

3 Ergebnisse

3.1 Kognitive Leistung

3.1.1 Anzahl richtiger Antworten

Die Anzahl richtiger Antworten betrug für null Distraktoren 98%, für zwei Distraktoren 95%, für vier Distraktoren 92% und für neun Distraktoren 73% (Abbildung 1).

3.1.1.1 MANOVA

In Tabelle 1 wird das Ergebnis der Multivariaten Testung (konfirmatorisch) über die Anzahl richtiger Antworten dargestellt.

Tabelle 1

Anzahl richtiger Antworten

df Effekt	MS Effekt	Df Fehler	MS Fehler	F	p-level
3	0,10	21	0,004	23,52	<0,001

3.1.1.2 Post-hoc-Tests

Die Anzahl richtiger Antworten ist bei null im Vergleich mit vier Distraktoren (0/4) und jeweils im Vergleich zum maximalen Schwierigkeitsgrad (0/9, 2/9, 4/9) signifikant größer.

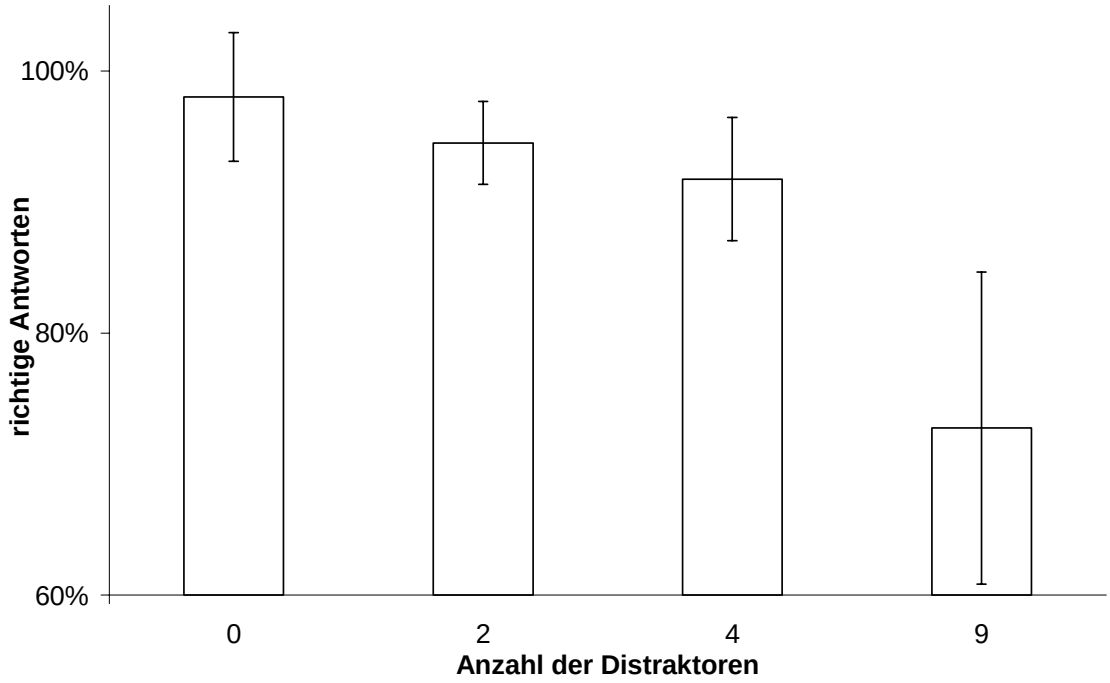
Tabelle 2

Anzahl richtiger Antworten

|t|	2	4	9
0	1.7	2.2*	5.0***
2		1.9	5.7***
4			5.8***

gepaarter zweiseitiger t-Test über die Anzahl richtiger Antworten
(Zeilen/Spalten: 0: 0 Distraktoren, 2: 2 Distraktoren, 4: 4 Distraktoren, 9: 9 Distraktoren)
* p<0.05; ** p<0.01, *** p<0.001

Abbildung 1: Richtige Antworten



3.1.2 Reaktionszeiten

Die Reaktionszeit betrug für null Distraktoren 595ms, für zwei Distraktoren 674ms, für vier Distraktoren 773ms und für neun Distraktoren 1062ms (Abbildung 2).

3.1.2.1 MANOVA

In Tabelle 2 wird das Ergebnis der Multivariaten Testung (konfirmatorisch) über die Reaktionszeiten dargestellt.

Tabelle 3

Reaktionszeiten

df Effekt	MS Effekt	df Fehler	MS Fehler	F	p-level
3	333117,45	21	11437,13	29,13	<0,001

3.1.2.2 Post-hoc-Tests

Die Reaktionszeit ist in jedem der möglichen Fälle (0/2, 0/4, 0/9, 2/4, 2/9, 4/9) bei größerer Distraktorenanzahl signifikant länger als bei kleinerer.

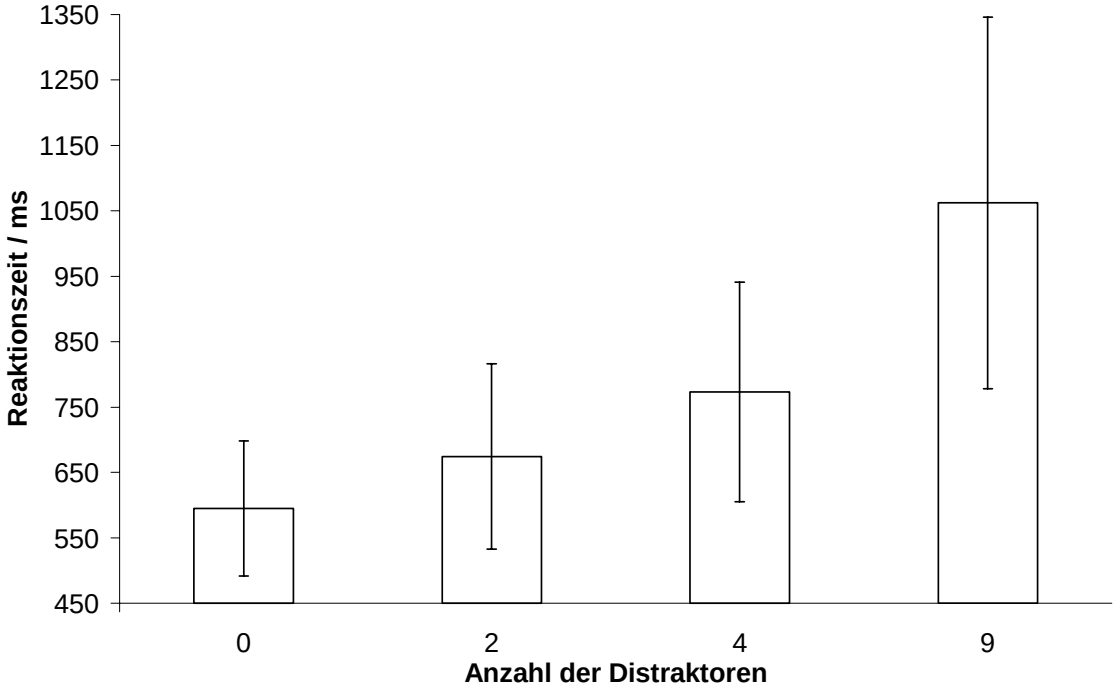
Tabelle 4

Reaktionszeit

|t|	2	4	9
0	3.9**	5.1***	5.9***
2		3.2**	5.4***
4			5.1***

gepaarter zweiseitiger t-Test über die Reaktionszeit
(Zeilen/Spalten: 0: 0 Distraktoren, 2: 2 Distraktoren, 4: 4 Distraktoren, 9: 9 Distraktoren)
* p<0.05; ** p<0.01, *** p<0.001

Abbildung 2: Reaktionszeiten



3.2 evozierte Potentiale

Nach visueller Inspektion des Kurvenverlaufs (Abbildung 3) wurde die Bereichseinteilung der EKP-Segmente, wie in Tabelle 5 dargestellt, gewählt.

Abbildung 3: Grand Average GFP (Brain-Maps: Verteilung der hirnelektrischen Aktivität in µV zum Zeitpunkt der Maxima der GFP der einzelnen Segmente)

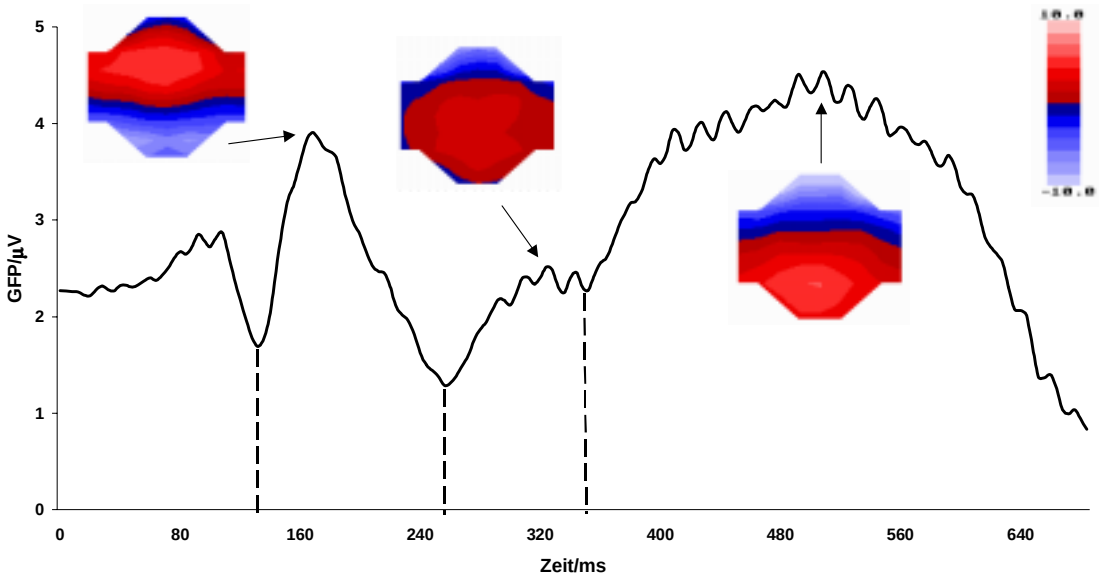


Tabelle 5

Bereichseinteilungen der Segmente

Segment Nummer	Start (ms)	Ende (ms)
1	64	128
2	132	256
3	260	348
4	352	688

linke Spalte: Nummer des Segments; mittlere Spalte: Start (ms) des Segments;
rechte Spalte: Ende (ms) des Segments

3.2.1 MANOVAs

In Tabelle 6 bis Tabelle 9 werden die Ergebnisse der Multivariaten Testung (konfirmatorisch) über die globale Feldstärke (GFP), die Latenz und die Kartentopographie (C-, C+, Cx, Cy, Distanz, Winkel) der EKP-Segmente dargestellt. Die Testung zeigt signifikante Zusammenhänge zwischen Segment und Distraktoren (12) auf für alle Parameter, wobei für den Zusammenhang zwischen Segment und Schwerpunkt noch ein p-Wert von 0,0503 als Trend angesehen wurde. Der Zusammenhang zwischen Segment und Trigger (F/T) ist nur signifikant für die GFP und den Schwerpunkt.

Tabelle 6

GFP

	Rao's R	df 1 (Effekt, Fehler)	df 2 (Effekt, Fehler)
1	2.3 *	6	62
2	2.3	2	31
3	0.4	2	31
12	5.1 *	6	62
13	2.7 *	6	62
23	6.2 *	2	31
123	1.0	6	62

Tabelle 7

Latenz

	Rao's R	df 1 (Effekt, Fehler)	df 2 (Effekt, Fehler)
1	71.7 *	6	62
2	3.7 *	2	31
3	0.0	2	31
12	9.4 *	6	62
13	1.1	6	62
23	1.0	2	31
123	2.0	6	62

Tabelle 8

Centroide (C-, C+)

	Rao's R	df 1 (Effekt, Fehler)	df 2 (Effekt, Fehler)
1	1.8 *	24	73
2	1.2	8	25
3	0.5	8	25
12	1.9 *	24	73
13	0.7	24	73
23	2.1	8	25
123	0.7	24	73

Tabelle 9

Schwerpunkt (Cx, Cy, Distanz, Winkel)

	Rao's R	df 1 (Effekt, Fehler)	df 2 (Effekt, Fehler)
1	1.0	24	73
2	1.1	8	25
3	0.9	8	25
12	1.7	24	73
13	1.7 *	24	73
23	4.2 *	8	25
123	1.2	24	73

Tabelle 6 - Tabelle 9:
4x4x2 MANOVA über die globale Feldstärke (GFP), die Latenz
und die Kartentopographie (C-, C+, Cx, Cy, Distanz, Winkel) der EKP-Segmente
(Faktoren: 1 Segment, 2 Distraktoren, 3 Trigger (F/T))
* p<0.05; ** p<0.01, *** p<0.001

3.2.2 Post-hoc-Tests

Im Folgenden werden in der MANOVA signifikante und durch Post-hoc-Tests bestätigte Ergebnisse dargestellt. Für solche Ergebnisse wird jeweils eine |t|-Tabelle angefügt.

3.2.2.1 Trigger F gegen T

Die GFP weist in keinem der vier Segmente signifikante Unterschiede zwischen den Triggern auf.
Der Schwerpunkt (Cx, Cy), Distanz (Dist) und Winkel (Angle) des Centroiddipols weisen bis auf eine Ausnahme in keinem der vier Segmente signifikante Unterschiede zwischen den Triggern auf: Im zweiten Segment ist der Schwerpunkt der Centroide bei einer Anzahl von zwei Distraktoren für Trigger F gegen Trigger T signifikant nach links verlagert.

Tabelle 10

Trigger F/T, Cx 2.Seg .

	|t|
0	0.1
2	3.5**
4	1.5
9	1.2

gepaarter zweiseitiger t-Test über die transversale Verlagerung des Schwerpunkts (Cx)
des 2. EKP-Segments für die Trigger F gegen T
(Zeilen: 0: 0 Distraktoren, 2: 2 Distraktoren, 4: 4 Distraktoren, 9: 9 Distraktoren)
* p<0.05; ** p<0.01, *** p<0.001

3.2.2.2 Distraktoren gegeneinander

3.2.2.2.1 Trigger F

Im ersten Segment ist die GFP signifikant niedriger bei weniger Distraktoren (0/2, 0/4, 0/9, 2/9, 4/9).

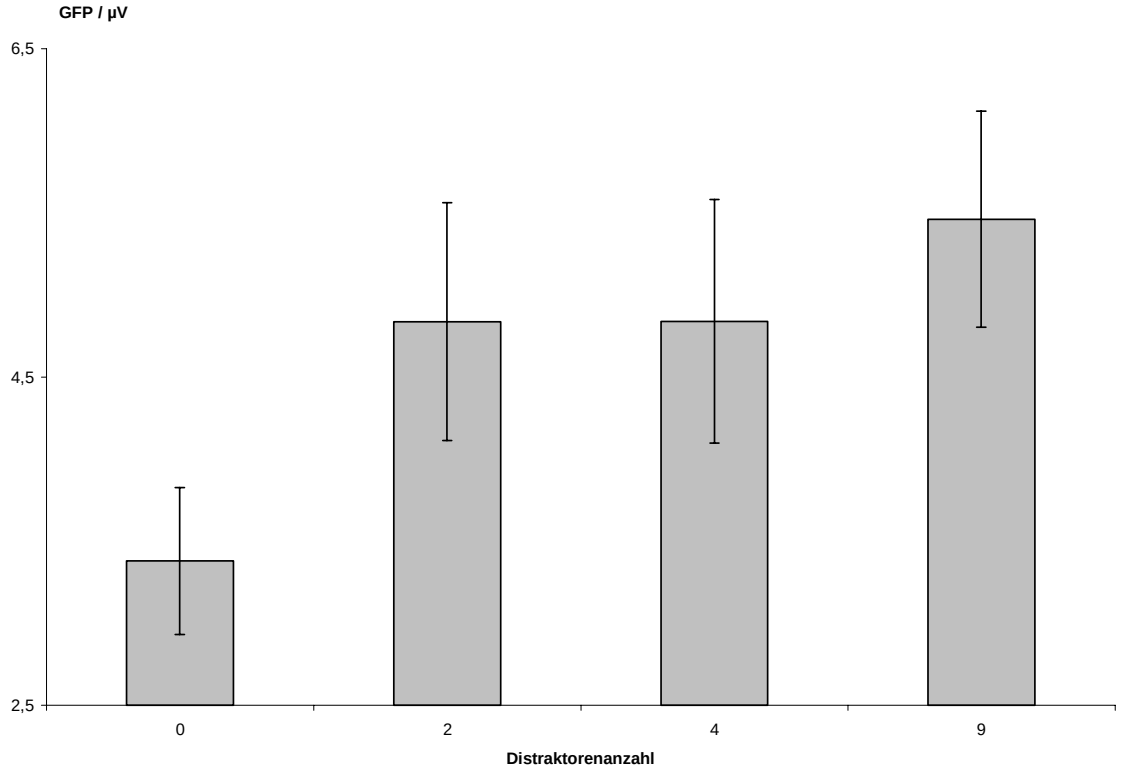
Tabelle 11

Trigger F, GFP 1.Seg .

|t|	2	4	9
0	3,4**	3,5**	5,3***
2		0,0	2,9*
4			2,3*

gepaarter zweiseitiger t-Test über die globale Feldstärke (GFP)
des 1. EKP-Segments für den Trigger F
(Zeilen/Spalten: 0: 0 Distraktoren, 2: 2 Distraktoren, 4: 4 Distraktoren, 9: 9 Distraktoren)
* p<0.05; ** p<0.01, *** p<0.001

Abbildung 4: Trigger F, Mittelwerte der Globalen Feldstärken im 1. Segment



Im zweiten Segment sieht man bei der GFP einen Trend, der aber nur zwischen null und neun Distraktoren (0/9) signifikant wird.

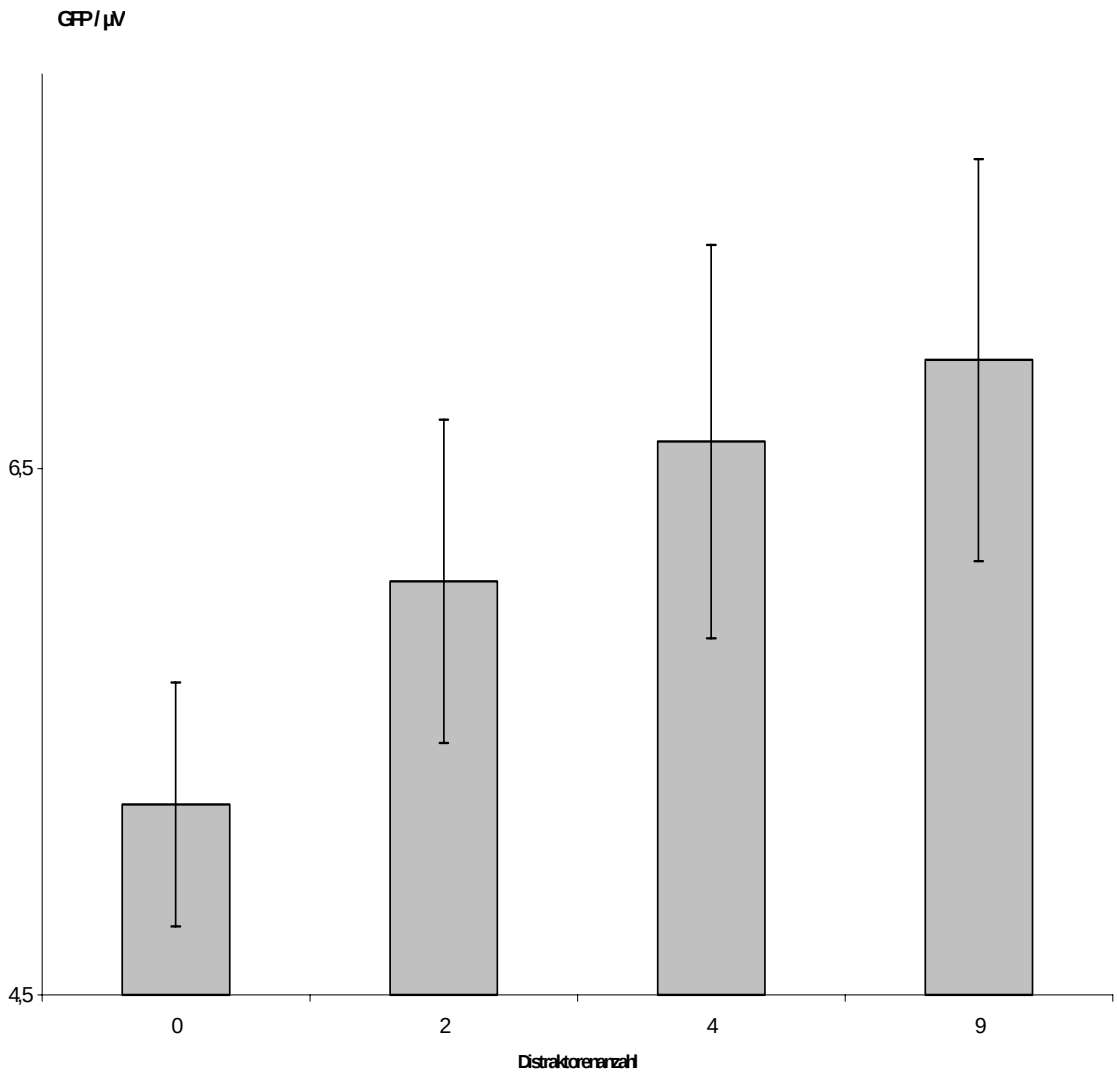
Tabelle 12

Trigger F, GFP 2.Seg .

|t|	2	4	9
0	1,2	2,1	2,2*
2		1,2	2,1
4			0,9

gepaarter zweiseitiger t-Test über die globale Feldstärke (GFP)
des 2. EKP-Segments für den Trigger F
(Zeilen/Spalten: 0: 0 Distraktoren, 2: 2 Distraktoren, 4: 4 Distraktoren, 9: 9 Distraktoren)
* p<0.05; ** p<0.01, *** p<0.001

Abbildung 5: Trigger F, Mittelwerte der Globalen Feldstärken im 2. Segment



Im dritten Segment ist kein signifikanter Unterschied der GFP bei verschiedener Distraktorenanzahl feststellbar. Im vierten Segment ist die GFP jeweils bei der Unterscheidung zum maximalen Schwierigkeitsgrad (0/9, 2/9, 4/9) signifikant höher.

Tabelle 13

Trigger F, GFP 4.Seg .

|t|	2	4	9
0	0,2	0,8	3,5**
2		0,5	8,0***
4			3,3**

gepaarter zweiseitiger t-Test über die globale Feldstärke (GFP)
des 4. EKP-Segments für den Trigger F
(Zeilen/Spalten: 0: 0 Distraktoren, 2: 2 Distraktoren, 4: 4 Distraktoren, 9: 9 Distraktoren)
* $p<0.05$; ** $p<0.01$, *** $p<0.001$

Abbildung 6: Trigger F, Mittelwerte der Globalen Feldstärken im 4. Segment

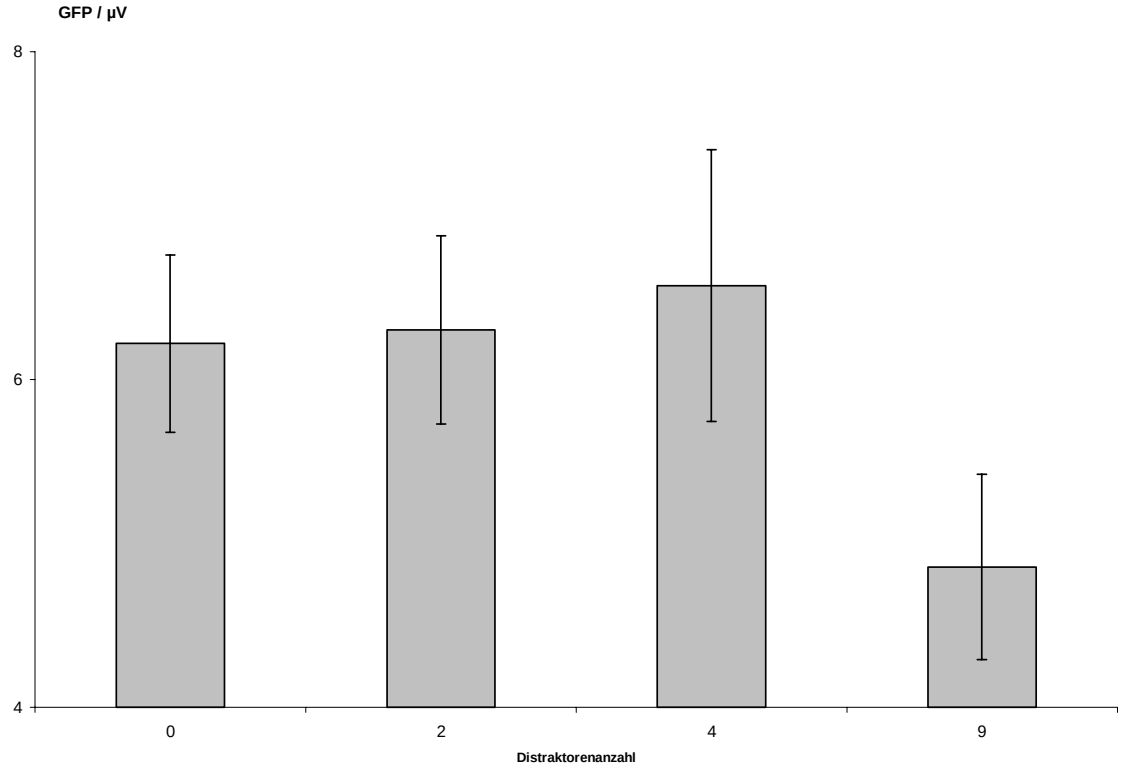
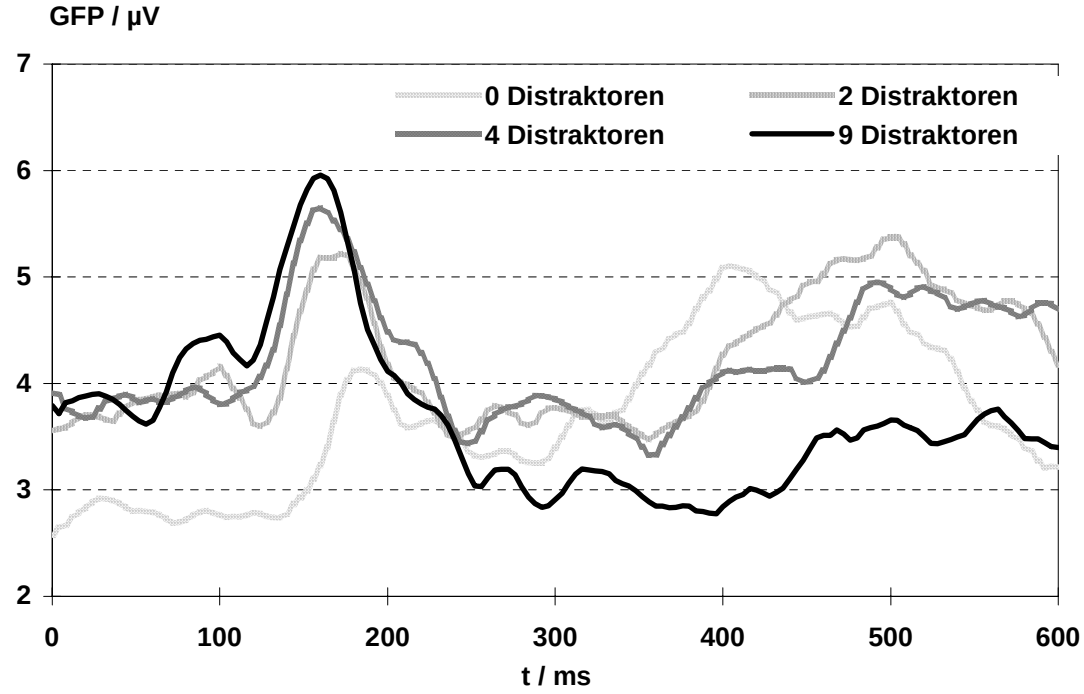


Abbildung 7: Trigger F, Globale Feldstärke



Im ersten und dritten Segment lässt sich für die Latenz kein signifikanter Unterschied bei verschiedener Distraktorenanzahl feststellen. Im zweiten Segment dagegen wird das GFP-Maximum bei weniger Distraktoren signifikant später erreicht (0/4, 0/9, 2/9).

Tabelle 14

Trigger F, Latenz 2.Seg .

|t|	2	4	9
0	1,4	2,4*	2,3*
2		0,0	3,3**
4			1,4

gepaarter zweiseitiger t-Test über die Latenz
des 2. EKP-Segments für den Trigger F
(Zeilen/Spalten: 0: 0 Distraktoren, 2: 2 Distraktoren, 4: 4 Distraktoren, 9: 9 Distraktoren)
* p<0.05; ** p<0.01, *** p<0.001

Im vierten Segment findet sich das GFP-Maximum bei weniger Distraktoren signifikant früher (0/2, 0/4, 0/9).

Tabelle 15

Trigger F, Latenz 4.Seg .

|t|	2	4	9
0	3,5**	5,1***	3,6**
2		2,0	0,7
4			0,9

gepaarter zweiseitiger t-Test über die Latenz des 4. EKP-Segments für den Trigger F
(Zeilen/Spalten: 0: 0 Distraktoren, 2: 2 Distraktoren, 4: 4 Distraktoren, 9: 9 Distraktoren)
* p<0.05; ** p<0.01, *** p<0.001

In den Segmenten eins, drei und vier ist keine signifikante transversale Verlagerung des negativen Centroids feststellbar (Cx1). Im zweiten Segment lässt sich für den negativen Centroid (Cx1) bei null gegen vier und zwei gegen vier Distraktoren (0/4, 2/4) eine signifikante Rechtsverlagerung feststellen.

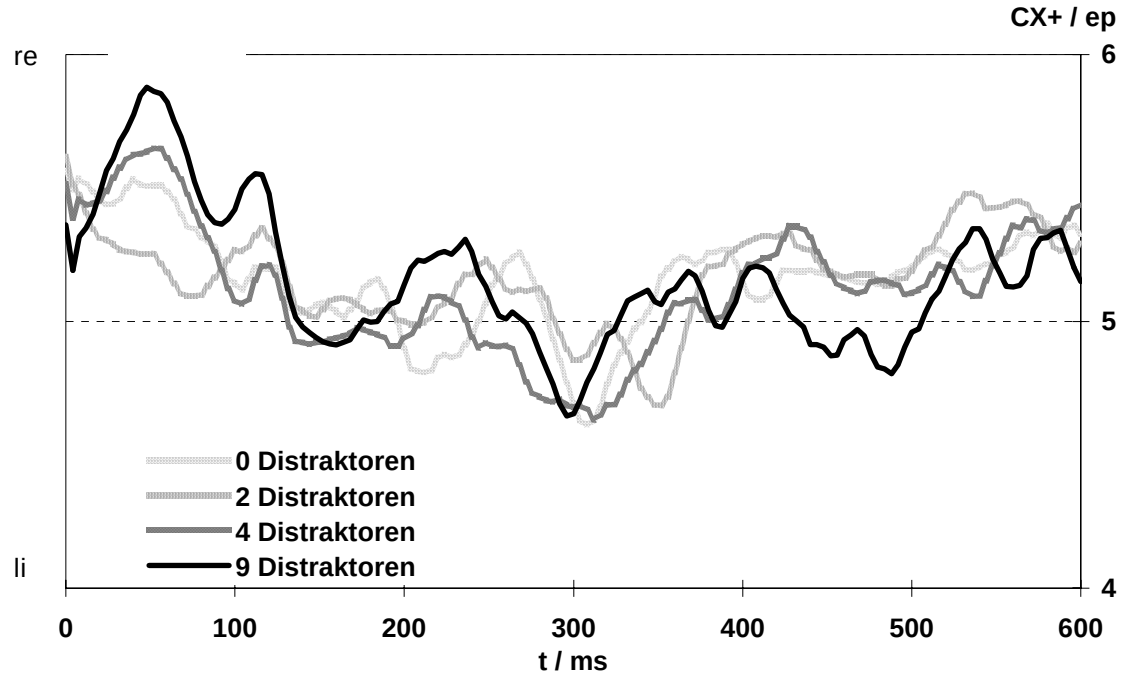
Tabelle 16

Trigger F, Cx- 2.Seg .

|t|	2	4	9
0	1,5	2,4*	0,1
2		4,3***	0,7
4			1,1

gepaarter zweiseitiger t-Test über die transversale Verlagerung des negativen Centroids (Cx-) des 2. EKP-Segments für den Trigger F
(Zeilen/Spalten: 0: 0 Distraktoren, 2: 2 Distraktoren, 4: 4 Distraktoren, 9: 9 Distraktoren)
* p<0.05; ** p<0.01, *** p<0.001

Abbildung 8: Trigger F, negativer Centroid (links-rechts)



Für den negativen Centroid (Cy1) ist in keinem der vier Segmente eine signifikante Verlagerung feststellbar. In den Segmenten eins, drei und vier ist keine signifikante transversale Verlagerung des positiven Centroids (Cx2) feststellbar. Im zweiten Segment lässt sich für den positiven Centroid (Cx2) bei zwei gegen neun Distraktoren (2/9) eine signifikante Rechtsverlagerung zeigen.

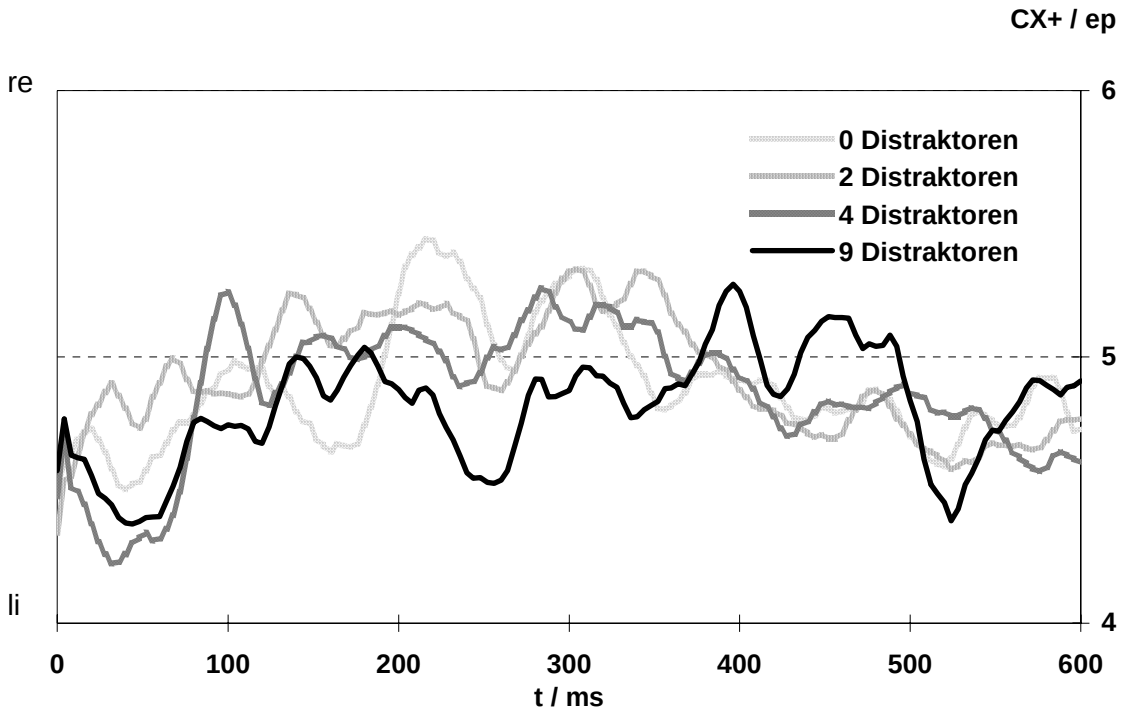
Tabelle 17

Trigger F, Cx+ 2.Seg .

|t|	2	4	9
0	0,5	0,2	1,4
2		0,3	2,2**
4			1,7

gepaarter zweiseitiger t-Test über die transversale Verlagerung des positiven Centroids (Cx+) des 2. EKP-Segments für den Trigger F
(Zeilen/Spalten: 0: 0 Distraktoren, 2: 2 Distraktoren, 4: 4 Distraktoren, 9: 9 Distraktoren)
* p<0.05; ** p<0.01, *** p<0.001

Abbildung 9: Trigger F, positiver Centroid (links-rechts)



Im ersten Segment lässt sich für den positiven Centroid (Cy2) bei null gegen zwei Distraktoren (0/2) eine signifikante Anteriorisierung feststellen,

Tabelle 18

Trigger F, Cy+ 1.Seg .

|t|	2	4	9
0	3,1***	0,4	0,2
2		2,0	1,4
4			0,4

gepaarter zweiseitiger t-Test über die sagittale Verlagerung des positiven Centroids (Cy+) des 1. EKP-Segments für den Trigger F
(Zeilen/Spalten: 0: 0 Distraktoren, 2: 2 Distraktoren, 4: 4 Distraktoren, 9: 9 Distraktoren)
* p<0.05; ** p<0.01, *** p<0.001

ebenso im 2. Segment bei null gegen vier Distraktoren (0/4).

Tabelle 19

Trigger F, Cy+ 2.Seg .

|t|	2	4	9
0	0,6	2,5**	0,6
2		1,2	0,1
4			1,2

gepaarter zweiseitiger t-Test über die sagittale Verlagerung des positiven Centroids (Cy+) des 2. EKP-Segments für den Trigger F
(Zeilen/Spalten: 0: 0 Distraktoren, 2: 2 Distraktoren, 4: 4 Distraktoren, 9: 9 Distraktoren)
* p<0.05; ** p<0.01, *** p<0.001

Im dritten Segment lässt sich für den positiven Centroid (Cy2) bei vier gegen neun Distraktoren (4/9) eine signifikante Posteriorisierung nachweisen.

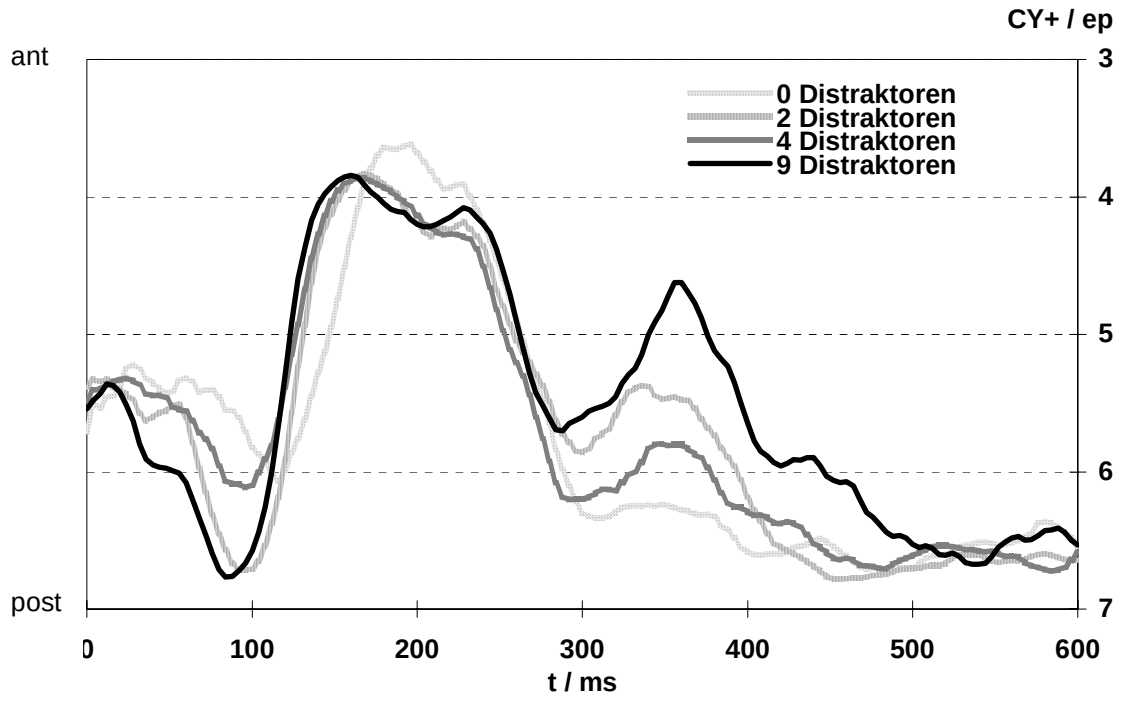
Tabelle 20

Trigger F, Cy+ 3.Seg .

|t|	2	4	9
0	0,9	0,4	1,7
2		1,6	0,7
4			2,2**

gepaarter zweiseitiger t-Test über die sagittale Verlagerung des positiven Centroids (Cy+) des 3. EKP-Segments für den Trigger F
(Zeilen/Spalten: 0: 0 Distraktoren, 2: 2 Distraktoren, 4: 4 Distraktoren, 9: 9 Distraktoren)
* p<0.05; ** p<0.01, *** p<0.001

Abbildung 10: Trigger F, positiver Centroid (anterior-posterior)



Im vierten Segment ist keine signifikante sagittale Verlagerung des positiven Centroids (Cy2) feststellbar. Im ersten Segment ist die Länge des Centroiddipols (Distanz) bei der Anzahl von null gegenüber zwei Distraktoren signifikant kürzer.

Tabelle 21

Trigger F, Distanz 1.Seg .

|t|	2	4	9
0	2,6*	1,3	1,9
2		1,1	0,4
4			0,7

gepaarter zweiseitiger t-Test über die Distanz
des 1. EKP-Segments für den Trigger F
(Zeilen/Spalten: 0: 0 Distraktoren, 2: 2 Distraktoren, 4: 4 Distraktoren, 9: 9 Distraktoren)
* p<0.05; ** p<0.01, *** p<0.001

Im Gegensatz dazu ist sie im zweiten Segment unter gleichen Bedingungen (0/2) signifikant länger.

Tabelle 22

Trigger F, Distanz 2.Seg .

|t|	2	4	9
0	3,0**	0,6	1,5
2		1,8	0,4
4			1,0

gepaarter zweiseitiger t-Test über die Distanz
des 2. EKP-Segments für den Trigger F
(Zeilen/Spalten: 0: 0 Distraktoren, 2: 2 Distraktoren, 4: 4 Distraktoren, 9: 9 Distraktoren)
* p<0.05; ** p<0.01, *** p<0.001

Im dritten Segment ist die Distanz bei der Gegenüberstellung von vier gegen neun Distraktoren (4/9) ebenfalls signifikant länger.

Tabelle 23

Trigger F, Distanz 3.Seg .

|t|	2	4	9
0	1,4	2,1	1,4
2		1,5	2,0
4			2,7*

gepaarter zweiseitiger t-Test über die Distanz
des 3. EKP-Segments für den Trigger F
(Zeilen/Spalten: 0: 0 Distraktoren, 2: 2 Distraktoren, 4: 4 Distraktoren, 9: 9 Distraktoren)
* p<0.05; ** p<0.01, *** p<0.001

Im vierten Segment ist die Distanz im Vergleich von null mit vier Distraktoren (0/4) signifikant kürzer.

Tabelle 24

Trigger F, Distanz 4.Seg .

|t|	2	4	9
0	1,8	3,0**	0,3
2		0,0	1,3
4			1,2

gepaarter zweiseitiger t-Test über die Distanz
des 4. EKP-Segments für den Trigger F
(Zeilen/Spalten: 0: 0 Distraktoren, 2: 2 Distraktoren, 4: 4 Distraktoren, 9: 9 Distraktoren)
* p<0.05; ** p<0.01, *** p<0.001

Der Winkel (Angle) des Centroiddipols weist in keinem der vier Segmente einen signifikanten Unterschied bei verschiedener Distraktorenanzahl auf. Der Schwerpunkt (Cx, Cy) des Centroiddipols zeigt im ersten Segment keine signifikante Verlagerung bei verschiedener Distraktorenanzahl. Im zweiten Segment ist der Schwerpunkt bei zwei im Vergleich mit vier (2/4) Distraktoren signifikant nach links verlagert (Cx).

Tabelle 25

Trigger F, Cx 2.Seg .

|t|	2	4	9
0	0,5	1,3	1,4
2		2,3*	2,1
4			0,1

gepaarter zweiseitiger t-Test über die transversale Verlagerung des Schwerpunkts (Cx)
des 2. EKP-Segments für den Trigger F
(Zeilen/Spalten: 0: 0 Distraktoren, 2: 2 Distraktoren, 4: 4 Distraktoren, 9: 9 Distraktoren)
* p<0.05; ** p<0.01, *** p<0.001

Außerdem findet sich in diesem Segment bei null im Vergleich mit vier (0/4) und bei null im Vergleich mit neun Distraktoren (0/9) eine signifikante Anteriorisierung (Cy).

Tabelle 26

Trigger F, Cy 2.Seg .

|t|	2	4	9
0	1,0	2,3*	3,4**
2		0,1	1,2
4			1,1

gepaarter zweiseitiger t-Test über die sagittale Verlagerung des Schwerpunkts (Cy)
des 2. EKP-Segments für den Trigger F
(Zeilen/Spalten: 0: 0 Distraktoren, 2: 2 Distraktoren, 4: 4 Distraktoren, 9: 9 Distraktoren)
* p<0.05; ** p<0.01, *** p<0.001

Der Schwerpunkt der Centroide weist weder im dritten noch im vierten Segment eine signifikante sagittale Verlagerung (Cy) bei verschiedener Distraktorenanzahl auf. Im dritten Segment zeigt sich bei der Gegenüberstellung von null mit zwei Distraktoren

(0/2) eine signifikante Rechtsverlagerung (Cx) und bei der Gegenüberstellung von zwei mit neun Distraktoren (2/9) eine signifikante Linksverlagerung (Cx) des Schwerpunkts.

Tabelle 27

Trigger F, Cx 3.Seg .

|t|	2	4	9
0	2,2*	0,1	0,8
2		1,4	2,8*
4			0,6

gepaarter zweiseitiger t-Test über die transversale Verlagerung des Schwerpunkts (Cx)
des 3. EKP-Segments für den Trigger F
(Zeilen/Spalten: 0: 0 Distraktoren, 2: 2 Distraktoren, 4: 4 Distraktoren, 9: 9 Distraktoren)
* p<0.05; ** p<0.01, *** p<0.001

Im vierten Segment liegt der Schwerpunkt im Vergleich von null mit zwei Distraktoren (0/2) signifikant weiter rechts (Cx).

Tabelle 28

Trigger F, Cx 4.Seg .

|t|	2	4	9
0	2,5*	0,5	0,6
2		1,6	0,7
4			0,2

gepaarter zweiseitiger t-Test über die transversale Verlagerung des Schwerpunkts (Cx)
des 4. EKP-Segments für den Trigger F
(Zeilen/Spalten: 0: 0 Distraktoren, 2: 2 Distraktoren, 4: 4 Distraktoren, 9: 9 Distraktoren)
* p<0.05; ** p<0.01, *** p<0.001

3.2.2.2.2 Trigger T

Im ersten Segment ist die GFP signifikant niedriger bei weniger Distraktoren (0/4, 0/9, 2/9, 4/9).

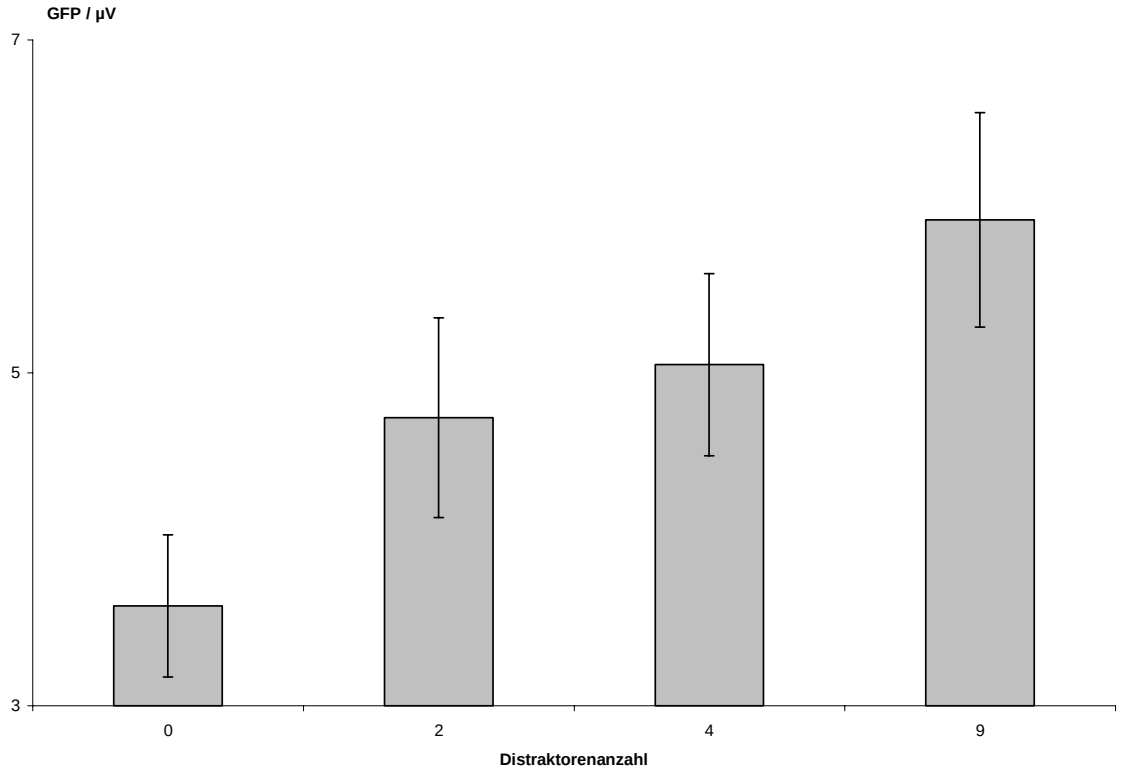
Tabelle 29

Trigger T, GFP 1.Seg .

|t|	2	4	9
0	1,9	3,4**	3,9**
2		0,7	2,3*
4			2,5*

gepaarter zweiseitiger t-Test über die globale Feldstärke (GFP)
des 1. EKP-Segments für den Trigger T
(Zeilen/Spalten: 0: 0 Distraktoren, 2: 2 Distraktoren, 4: 4 Distraktoren, 9: 9 Distraktoren)
* p<0.05; ** p<0.01, *** p<0.001

Abbildung 11: Trigger T, Mittelwerte der Globalen Feldstärken im 1. Segment



Im zweiten Segment ist die GFP jeweils bei der Unterscheidung zum minimalen Schwierigkeitsgrad (0/2, 0/4, 0/9) signifikant niedriger.

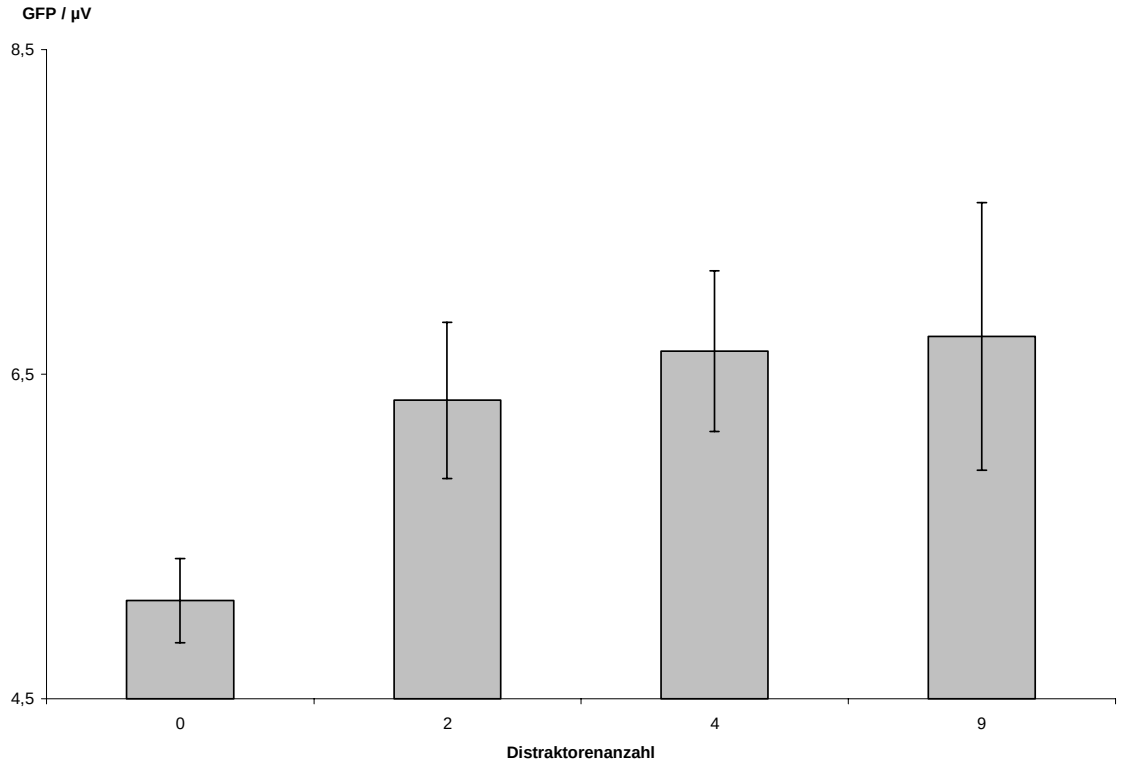
Tabelle 30

Trigger T, GFP 2.Seg .

|t|	2	4	9
0	2,9*	3,8**	2,3*
2		0,8	0,7
4			0,2

gepaarter zweiseitiger t-Test über die globale Feldstärke (GFP)
des 2. EKP-Segments für den Trigger T
(Zeilen/Spalten: 0: 0 Distraktoren, 2: 2 Distraktoren, 4: 4 Distraktoren, 9: 9 Distraktoren)
* p<0.05; ** p<0.01, *** p<0.001

Abbildung 12: Trigger T, Mittelwerte der Globalen Feldstärken im 2. Segment



Im dritten Segment ist die GFP zwischen keinem und vier (0/4) und keinem und neun Distraktoren (0/9) signifikant höher.

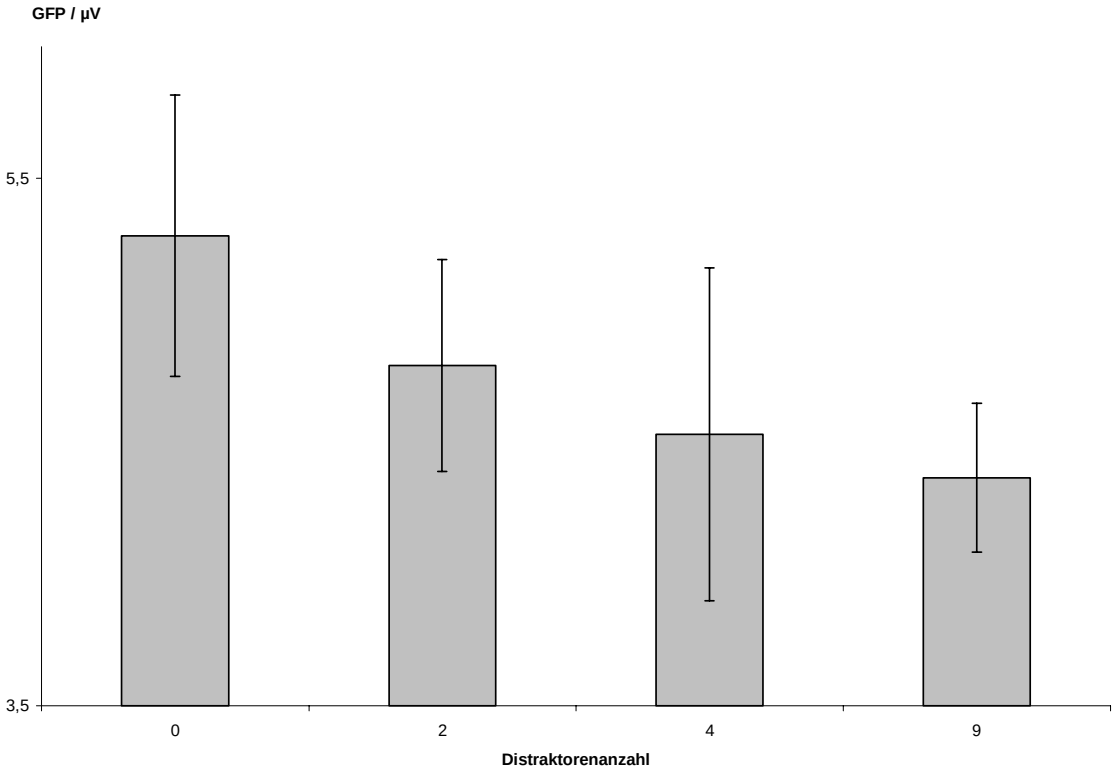
Tabelle 31

Trigger T, GFP 3.Seg .

|t|	2	4	9
0	1,3	2,3*	2,2*
2		0,8	1,1
4			0,3

gepaarter zweiseitiger t-Test über die globale Feldstärke (GFP)
des 3. EKP-Segments für den Trigger T
(Zeilen/Spalten: 0: 0 Distraktoren, 2: 2 Distraktoren, 4: 4 Distraktoren, 9: 9 Distraktoren)
* $p<0.05$; ** $p<0.01$, *** $p<0.001$

Abbildung 13: Trigger T, Mittelwerte der Globalen Feldstärken im 3. Segment



Auch im vierten Segment ist sie signifikant höher zwischen null und neun (0/9) Distraktoren.

Tabelle 32

Trigger T, GFP 4.Seg .

|t|	2	4	9
0	0,4	0,3	2,3*
2		0,2	1,9
4			1,4

gepaarter zweiseitiger t-Test über die globale Feldstärke (GFP)
des 4. EKP-Segments für den Trigger T
(Zeilen/Spalten: 0: 0 Distraktoren, 2: 2 Distraktoren, 4: 4 Distraktoren, 9: 9 Distraktoren)
* $p<0.05$; ** $p<0.01$, *** $p<0.001$

Abbildung 14: Trigger T, Mittelwerte der Globalen Feldstärken im 4. Segment

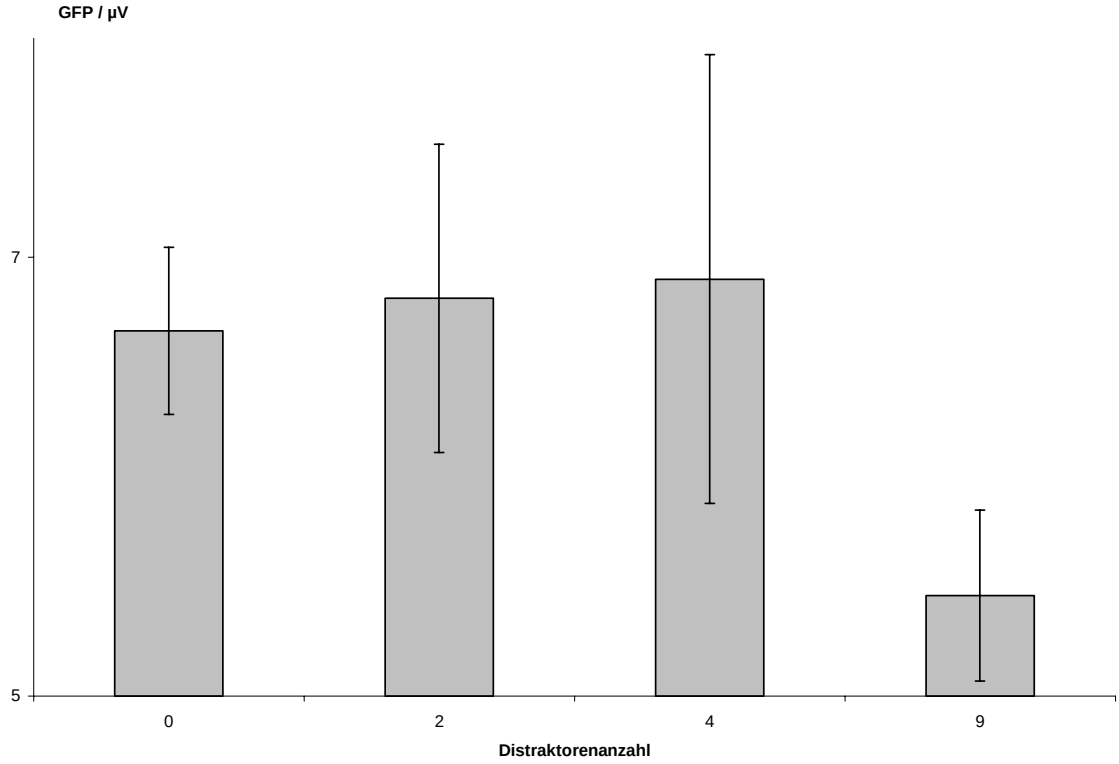
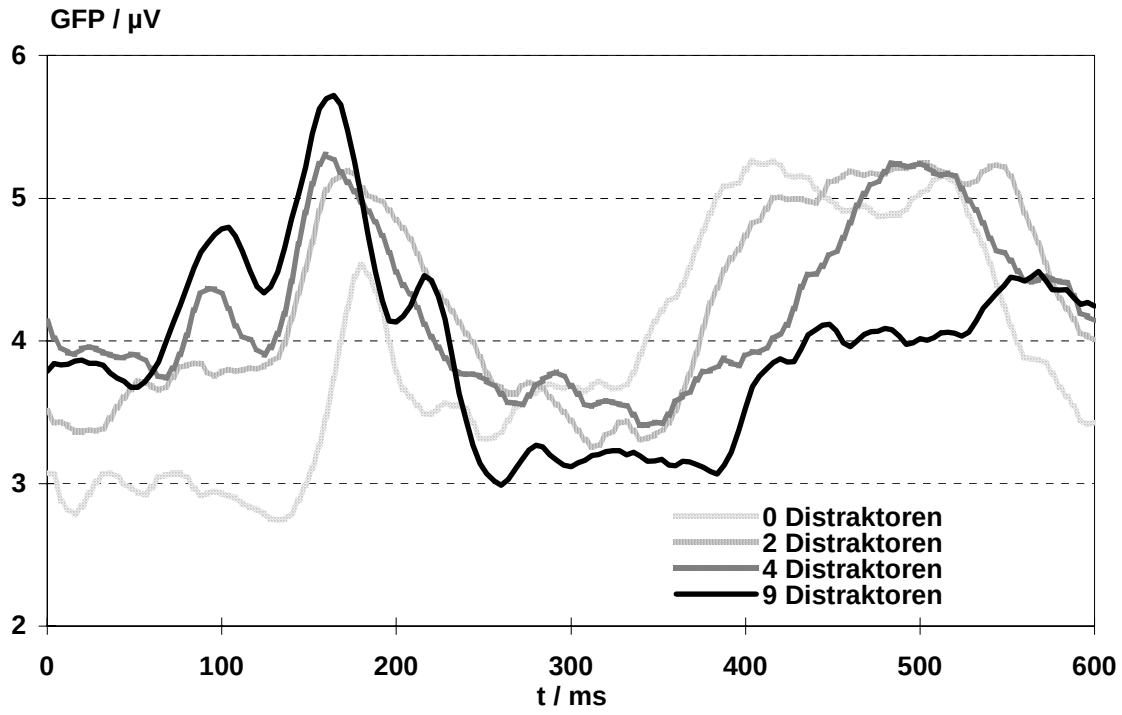


Abbildung 15: Trigger T, Globale Feldstärke



Im ersten Segment lässt sich für die Latenz kein signifikanter Unterschied bei verschiedener Distraktorenanzahl feststellen. Im zweiten Segment findet sich das GFP-Maximum bei null gegen neun (0/9) und zwei gegen neun (2/9) Distraktoren signifikant später.

Tabelle 33

Trigger T, Latenz 2.Seg .

|t|	2	4	9
0	0,9	0,9	2,5*
2		0,2	2,4*
4			1,7

gepaarter zweiseitiger t-Test über die Latenz
des 2. EKP-Segments für den Trigger T
(Zeilen/Spalten: 0: 0 Distraktoren, 2: 2 Distraktoren, 4: 4 Distraktoren, 9: 9 Distraktoren)
* $p<0.05$; ** $p<0.01$, *** $p<0.001$

Im dritten Segment ist die Latenz signifikant später zwischen null und vier Distraktoren (0/4).

Tabelle 34

Trigger T, Latenz 3.Seg .

|t|	2	4	9
0	0,6	2,4*	1,3
2		1,7	1,8
4			1,5

gepaarter zweiseitiger t-Test über die Latenz des 3. EKP-Segments für den Trigger T
(Zeilen/Spalten: 0: 0 Distraktoren, 2: 2 Distraktoren, 4: 4 Distraktoren, 9: 9 Distraktoren)
* p<0.05; ** p<0.01, *** p<0.001

Im vierten Segment ist die Latenz ist signifikant früher zwischen null und vier Distraktoren (0/4).

Tabelle 35

Trigger T, Latenz 4.Seg .

|t|	2	4	9
0	0,9	2,1*	2,0
2		1,7	0,4
4			0,5

gepaarter zweiseitiger t-Test über die Latenz
des 4. EKP-Segments für den Trigger T
(Zeilen/Spalten: 0: 0 Distraktoren, 2: 2 Distraktoren, 4: 4 Distraktoren, 9: 9 Distraktoren)
* p<0.05; ** p<0.01, *** p<0.001

In keinem der vier Segmente lässt sich weder für die Position des negativen noch für die des positiven Centroids eine signifikante transversale Verlagerung (Cx1, Cx2) feststellen. Im ersten Segment lässt sich für den negativen Centroid bei weniger Distraktoren (0/2, 0/4, 0/9) eine signifikante Posteriorisierung (Cy1) feststellen.

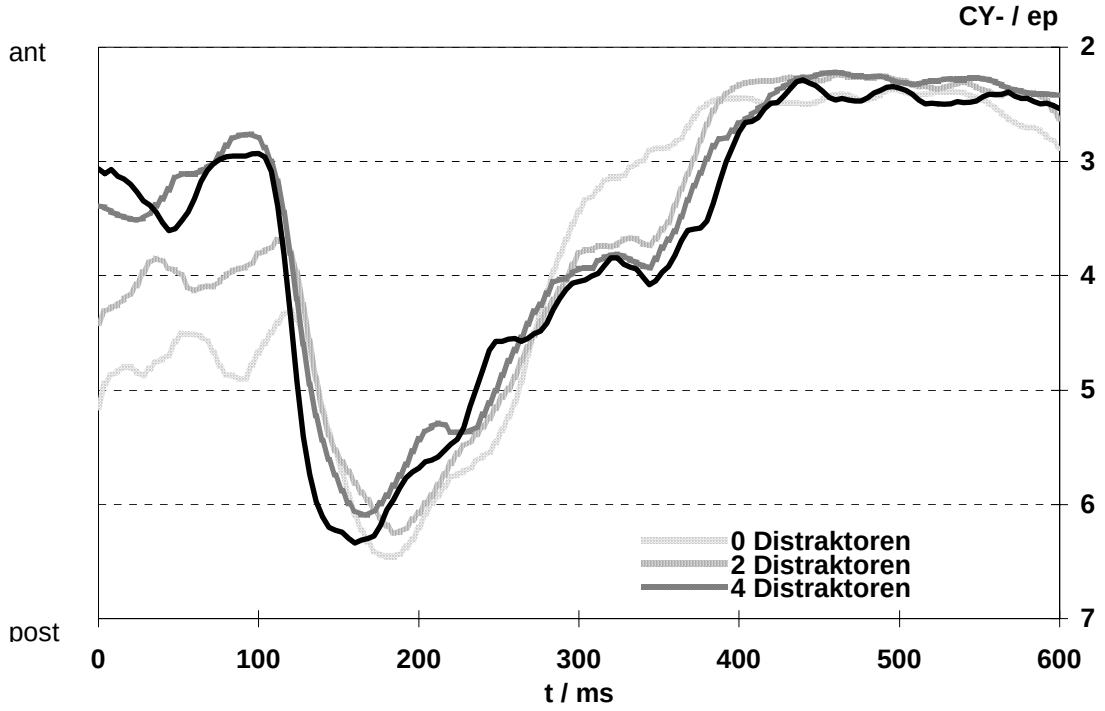
Tabelle 36

Trigger T, Cy- 1.Seg .

|t|	2	4	9
0	2,1*	3,4**	3,1**
2		1,5	0,8
4			0,9

gepaarter zweiseitiger t-Test über die sagittale Verlagerung des negativen Centroids (Cy-)
des 1. EKP-Segments für den Trigger T
(Zeilen/Spalten: 0: 0 Distraktoren, 2: 2 Distraktoren, 4: 4 Distraktoren, 9: 9 Distraktoren)
* p<0.05; ** p<0.01, *** p<0.001

Abbildung 16: Trigger T, negativer Centroid (anterior-posterior)



Für den positiven Centroid lässt sich im ersten Segment bei weniger Distraktoren (0/2, 0/4, 0/9) eine signifikante Anteriorisierung (Cy2) feststellen.

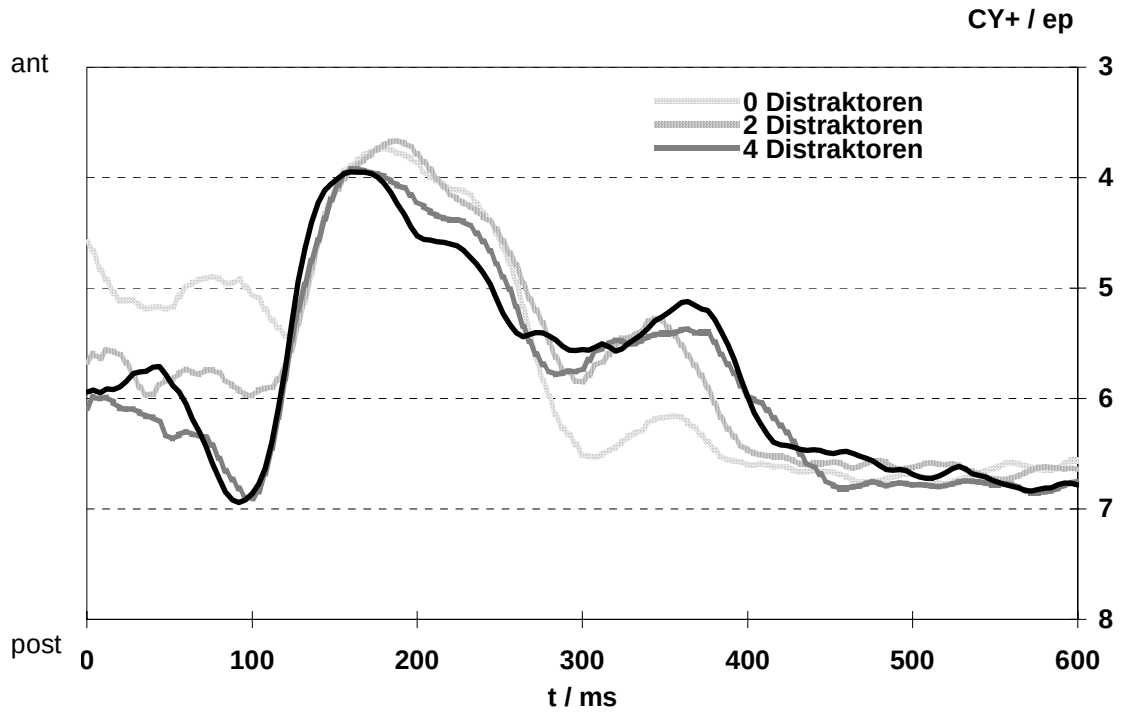
Tabelle 37

Trigger T, Cy+ 1.Seg .

|t|	2	4	9
0	2,8*	4,6***	5,1***
2		1,8	1,4
4			0,6

gepaarter zweiseitiger t-Test über die sagittale Verlagerung des positiven Centroids (Cy+) des 1. EKP-Segments für den Trigger T
(Zeilen/Spalten: 0: 0 Distraktoren, 2: 2 Distraktoren, 4: 4 Distraktoren, 9: 9 Distraktoren)
* p<0.05; ** p<0.01, *** p<0.001

Abbildung 17: Trigger T, positiver Centroid (anterior-posterior)



In den Segmenten zwei bis vier lässt sich für die Position des negativen und des positiven Centroids keine signifikante sagittale Verlagerung (Cy1, Cy2) feststellen.

Im ersten Segment ist die Länge des Centroiddipols (Distanz) bei geringstem Schwierigkeitsgrad gegen jegliche Distraktorenanzahl (0/2, 0/4, 0/9) signifikant kürzer.

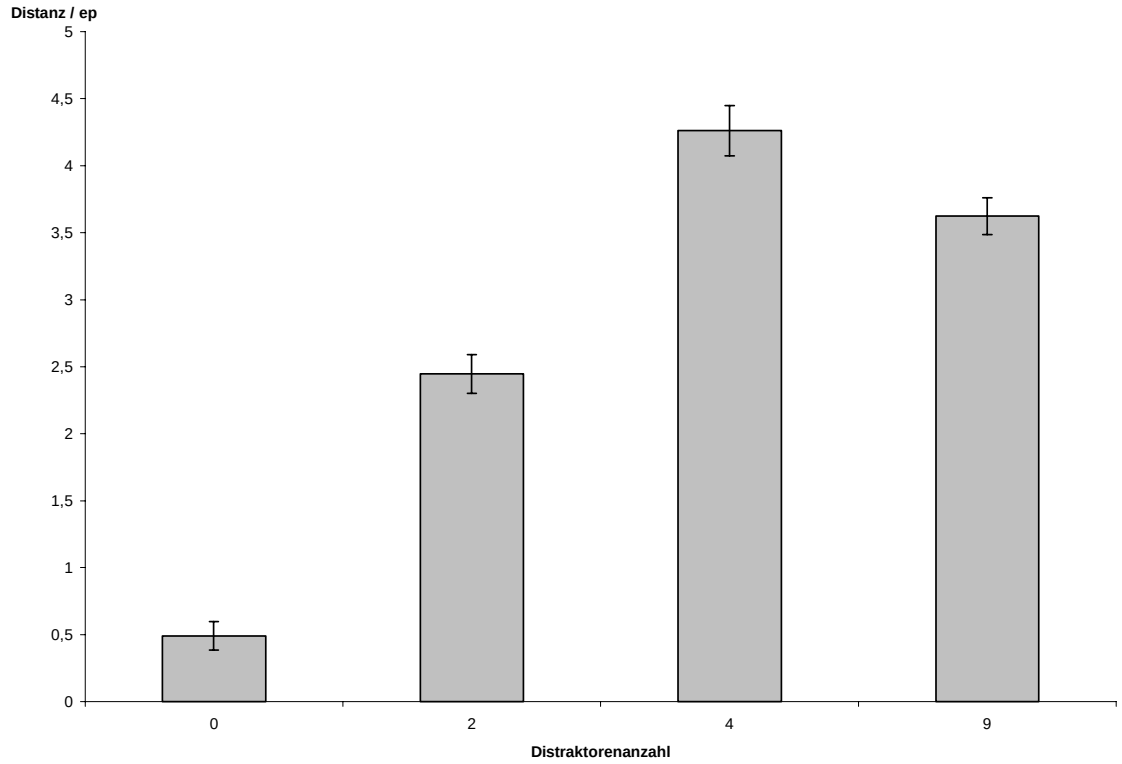
Tabelle 38

Trigger T, Distanz 1.Seg .

|t|	2	4	9
0	2,5*	3,7**	3,3**
2		2,1	0,6
4			0,8

gepaarter zweiseitiger t-Test über die Distanz
des 1. EKP-Segments für den Trigger T
(Zeilen/Spalten: 0: 0 Distraktoren, 2: 2 Distraktoren, 4: 4 Distraktoren, 9: 9 Distraktoren)
* p<0.05; ** p<0.01, *** p<0.001

Abbildung 18: Trigger T, Distanzen im Segment 1



Im zweiten Segment weist die Distanz keine signifikante Längenänderung auf. Im dritten Segment ist die Distanz bei der Gegenüberstellung von null gegen neun und vier gegen neun Distraktoren (0/9, 4/9) signifikant länger.

Tabelle 39

Trigger T, Distanz 3.Seg .

|t|	2	4	9
0	1,3	1,7	2,6*
2		0,6	1,6
4			2,2*

gepaarter zweiseitiger t-Test über die Distanz
des 3. EKP-Segments für den Trigger T
(Zeilen/Spalten: 0: 0 Distraktoren, 2: 2 Distraktoren, 4: 4 Distraktoren, 9: 9 Distraktoren)
* p<0.05; ** p<0.01, *** p<0.001

Im vierten Segment ist die Distanz bei der Gegenüberstellung von null gegen vier und zwei gegen vier Distraktoren (0/4, 2/4) signifikant kürzer.

Tabelle 40

Trigger T, Distanz 4.Seg .

|t|	2	4	9
0	1,2	2,7*	0,7
2		2,9*	0,0
4			1,4

gepaarter zweiseitiger t-Test über die Distanz
des 4. EKP-Segments für den Trigger T
(Zeilen/Spalten: 0: 0 Distraktoren, 2: 2 Distraktoren, 4: 4 Distraktoren, 9: 9 Distraktoren)
* p<0.05; ** p<0.01, *** p<0.001

Der Schwerpunkt (Cx, Cy) und Winkel (Angle) des Centroiddipols weisen bis auf eine Ausnahme in keinem der vier Segmente einen signifikanten Unterschied bei verschiedener Distraktorenanzahl auf: Im zweiten Segment ist der Schwerpunkt bei zwei im Vergleich mit vier Distraktoren (2/4) signifikant nach rechts verlagert (Cx).

Tabelle 41

Trigger T, Cx 2.Seg .

|t|	2	4	9
0	1,4	0,7	0,5
2		2,1*	1,3
4			0,0

gepaarter zweiseitiger t-Test über die transversale Verlagerung des Schwerpunkts (Cx)
des 2. EKP-Segments für den Trigger T
(Zeilen/Spalten: 0: 0 Distraktoren, 2: 2 Distraktoren, 4: 4 Distraktoren, 9: 9 Distraktoren)
* p<0.05; ** p<0.01, *** p<0.001

4 Diskussion

4.1 Kognitive Leistung

4.1.1 Anzahl richtiger Antworten

Betrachtet wird jeweils die Summe der richtigen Antworten, die von den Probanden bei dem jeweiligen Schwierigkeitsgrad erreicht wird. Postuliert wird zuvor eine Zunahme des Schwierigkeitsgrades und damit weniger zu erwartende richtige Antworten bei steigender Distraktorenzahl. Diese umgekehrte Proportionalität von erbrachter Leistung und der Anzahl der Antwortmöglichkeiten wird bereits 1966 von Warrington festgestellt [45]. Sie wird von vielen weiteren Studien bestätigt, die den Span-of-Apprehension-Test in ähnlicher Weise wie in der vorliegenden Studie durchführen [1;30]. Auch die einen mit der vorliegenden Studie nahezu identischen Versuchsaufbau verwendeten neueren Studien, die ebenfalls Personalcomputer zur Testdurchführung einsetzen, kommen zu diesem Ergebnis [5;6]. Mit der vorliegenden Studie kann dieser Sachverhalt bestätigt werden. Er spiegelt sich deutlich in Abbildung 1 auf Seite 10 wieder.

4.1.2 Reaktionszeiten

Die oben dargelegten Zusammenhänge finden sich auch für die Reaktionszeiten. In den bereits zitierten Publikationen werden wie in der vorliegenden Studie längere Reaktionszeiten bei größeren Distraktorenzahlen und somit höherem Schwierigkeitsgrad beobachtet. Abbildung 2 auf Seite 11 veranschaulicht dies.

4.2 Evozierte Potentiale

4.2.1 Trigger F gegen T

Zwischen der neuronalen Verarbeitung eines erkannten "F" und eines erkannten "T" und dem damit verbundenen Betätigen der linken oder der rechten Maustaste ließen sich in der Ausprägung der EKP-Deskriptoren mit einer Ausnahme keine signifikanten Unterschiede feststellen (3.2.2.1, Seite 14). In der Literatur gibt keine Hinweise auf Unterschiede in der Verarbeitung einzelner Buchstaben des lateinischen Alphabets, welche mittels neurophysiologischer Methoden, wie in dieser Studie angewandt, nachgewiesen werden könnten. Bei der gefundenen Abweichung im Bereich des Centroidenschwerpunktes handelt es sich eher um einen Zufallsbefund, da er nur in einem Microstate und nur bei 2 Distraktoren auftritt.

4.2.2 Distraktoren gegeneinander

Im Folgenden werden Zeiträume nach dem Stimulus durch Millisekunden oder Segmentnummern gekennzeichnet. Eine genaue Zuordnung ist Tabelle 5 auf Seite 12 zu entnehmen.

4.2.2.1 Latenzen

Es zeigt sich, dass das GFP-Maximum im zweiten Segment, entsprechend ca. 150 ms nach dem Ereignis, bei weniger Distraktoren, also leichterer kognitiver Anforderung, mit einer größeren Latenz eintritt (3.2.2.2, Seite 14). Dies trifft sowohl für Trigger F (Abbildung 7, Seite 18) als auch Trigger T (Abbildung 15, Seite 30) zu.

Positive Potentiale 150 ms nach dem Stimulus werden in der Literatur als P150 oder p2 bezeichnet. Die Latenz der P2 ist abhängig davon, ob ein Ereignis erwartet wird [15].

Sie fällt länger aus, wenn auf das Ereignis zuvor hingewiesen wird [46]. Sie fällt um so kürzer aus je deutlicher (größer) der Stimulus ist [20]. Aus der oben beschriebenen größeren Latenz bei geringerer Distraktorenanzahl kann gefolgert werden, dass bei Information, für deren Verarbeitung eine geringere Aufmerksamkeit benötigt wird, die Aufmerksamkeit weniger schnell aufgebracht wird als bei Information, deren Verarbeitung hoher Konzentration bedarf. Komplexerer Information scheint die Aufmerksamkeit schneller bereitgestellt zu werden.

Ganz entgegengesetzt lässt sich ca. 400 ms nach dem Stimulus im vierten Segment feststellen, dass die GFP-Maxima bei weniger Distraktoren, also leichterer kognitiver Anforderung, mit früher eintreten (3.2.2.2, Seite 14). Dies trifft wieder sowohl für Trigger F (Abbildung 7, Seite 18) als auch Trigger T (Abbildung 15, Seite 30) zu.

Positive Potentiale 400 ms nach dem Stimulus können als späte P300 bezeichnet werden. Die P300 wird durch eine Steigerung der Gedächtnisarbeitslast (engl.: working memory load) beeinflusst [28].

4.2.2.2 GFP-Mittelwerte

Für beide Trigger finden sich bei allen Schwierigkeitsgraden im ersten Segment, ca. 100 ms nach dem Ereignis, positive Potentiale. Die P100 widerspiegelt die visuelle Stimulation. Die Tatsache, dass bei höherer Distraktorenzahl höhere Amplituden der P100 gefunden werden, ist mit der höheren Leuchtstärke der Stimulation zu erklären [27].

Im zweiten Segment findet sich bei kleinerer Distraktorenanzahl eine niedrigere mittlere globale Feldstärke. Dies zeigt sich deutlich für Trigger F (Abbildung 4, Seite 15) und Trigger T (Abbildung 11, Seite 26). Aus der obigen Beobachtung kann man schlussfolgern, dass einfachere Information einen geringeren Grad an Aufmerksamkeit benötigen als komplexere. Aufwendiger zu verarbeitender Information wird auch mit einer größeren Aufmerksamkeit begegnet, hier repräsentiert durch die hohe Amplitude der p150, d.h. es werden mehr Neuronenverbände aktiviert, um die Aufgabe zu bewältigen.

Im vierten Segment findet sich bei größerer Distraktorenanzahl eine niedrigere GFP. Auch dies zeigt sich deutlich für Trigger F (Abbildung 6, Seite 18) und Trigger T (Abbildung 14, Seite 29).
Bei einem visuellen Gedächtnistest zeigt sich bei einem höheren Schwierigkeitsgrad eine geringere P300 [16]. Ein vergleichbares Ergebnis wurde bei einem Arbeitsgedächtnistest bei gesteigerter Leistungsanforderung festgestellt [14].

4.2.2.3 Zusammenfassende Diskussion zu Latenzen und GFP-Mittelwerte

Aus den oben beschriebenen Veränderungen der P300 in dieser Studie lässt sich ableiten, dass der Verarbeitungsprozess von einfacher zu verarbeitender Information früher abgeschlossen wird als von komplexer Information. Bei hohem Schwierigkeitsgrad wird die späte P300 durch die große Arbeitslast bedingt, die frühe p150 durch die hohe zu erbringende Leistung. Bei niedrigem Schwierigkeitsgrad erzeugt die geringe „working load“ eine frühe p300, die niedrige zu erbringende Leistung eine späte p150.
Dieses Phänomen scheint zunächst im Widerspruch mit dem im zweiten Segment beschriebenen zu stehen, lässt sich aber wie folgt erklären: Mit zunehmendem Schwierigkeitsgrad und folglich zunehmender Fehlerhäufigkeit nimmt bei entsprechenden kognitiven Prozessen die Effizienz ab und die Fehlerhaftigkeit zu. Dies spiegelt sich in der reduzierten Amplitude der späten kognitiven Potentiale wieder; dies kann als Ausdruck der Zunahme fehlerhafter neuronaler Aktivierungen bzw. stärker variierender neuronaler Aktivationsmuster beim Versuch der Bewältigung der Aufgabe gedeutet werden. Bildlich formuliert induziert eine höhere Arbeitslast in jedem Einzeltrial ein anderes Neuronenaktivationsmuster, da verschiedene Lösungswege probiert werden; bei der Mittelung der Ereignisse überlagern sich so die Maxima der Einzeltrials nicht exakt und es entsteht eine weniger scharf abgrenzbare p300. Weniger komplexe Information hingegen erzeugt während jedem Einzeltrial ein stabiles Aktivationsmuster, die dann im Mittel durch Überlagerung eine größere Amplitude der späten p300 ergeben

4.2.2.4 Centroide

Das Ergebnis ist wieder für F und T sehr ähnlich. Bei beiden Triggern zeigt sich im ersten Segment eine Posteriorisierung des negativen Centroids (Abbildung 16, Seite 32) bei gleichzeitiger Anteriorisierung des positiven Centroids (Abbildung 10, Seite 22 und Abbildung 17, Seite 33), wenn weniger Distraktoren gemeinsam mit den Triggern in der Matrix erscheinen. Anders formuliert entfernen sich negativer und positiver Centroid zu einem gleichen Zeitpunkt in den ersten 128 Millisekunden um so weiter voneinander, je niedriger die Anforderung an die Informationsverarbeitung ist. Der positive Centroid entfernt sich in Richtung Nase, der negative in Richtung Hinterhaupt.

Während die Bewegung der Centroide zu verschiedenen Zeitpunkten auf der vertikalen Achse (anterior-posterior) sehr gleichmäßig abläuft, ist sie auf der horizontalen Achse (links-rechts) größeren Schwankungen unterworfen. In der horizontalen Ausrichtung, die in ihrer Auslenkung deutlich geringer ist als die sagittale, lässt sich anhand des Kurvenverlaufs nur ein an verschiedenen Schwierigkeitsgraden orientierter Zusammenhang erahnen; es werden jedoch auch hier signifikante Unterschiede in der Centroidbewegung bei unterschiedlich hoher Anforderung gefunden (Abbildung 8, Seite 20 und Abbildung 9, Seite 21).

Die Länge des Centroiddipols, gemessen als Distanz, ist sowohl für Trigger F als auch für Trigger T im ersten und vierten Segment bei geringer Anforderung kleiner als bei hoher Anforderung. Im zweiten und dritten Segment ist die Distanz bei geringerer aufzuwendender kognitiver Leistung länger als bei hoher.

Die im ersten Segment festgestellte Verkürzung der Distanz bei geringerer Anforderung steht in Einklang mit der im selben Segment oben beschriebenen vertikalen Bewegungsrichtung der Centroide: Bei abnehmender Komplexizität der Information bewegen sich die beiden Centroide, der negative von anterior und der positive von posterior kommend, aufeinander zu. Eine anatomische Abklärung dieses Phänomens könnte mit einer Methode zur Quellenlokalisation wie z.B. LORETA [31] angegangen werden.

Es ist bekannt, dass bei tiefer liegenden hirnelektrischen Quellen die Distanz des Dipols auf der Schädeloberfläche (d.h. der Abstand der Centroide) größer ist als bei oberflächlichen Quellen. Die Distanz wird allerdings auch von der Ausdehnung der anderen aktiven Neuronenverbände beeinflusst, so dass eine verbindliche Aussage ohne Modellberechnung nicht möglich ist.

Signifikante Unterschiede für die Winkel des Centroiddipols bei unterschiedlicher Schwierigkeit konnten nicht festgestellt werden. Die Ergebnisse über die Centroidschwerpunktbewegung weisen in keine einheitliche Richtung

5 Schlussfolgerungen

Bei einer Gruppe von 9 gesunden Probanden wurde während einer EEG-Ableitung mit 64 Kanälen der Span-of-Apprehension-Test durchgeführt. Auf der Verhaltensebene zeigte sich mit Zunahme des Schweregrades des SAT, d.h. mit Zunahme der Distraktorenanzahl, eine Verlängerung der Reaktionszeiten sowie eine Zunahme der Fehlerrate. Weder auf der Verhaltensebene noch bei den evozierten Potentialen fand sich ein Unterschied zwischen Zielbuchstaben F oder T.

Eine erhöhte Anforderung an die Probanden während des SAT spiegelt sich in den evozierten Potentialen wie folgt wieder: Die frühen Potentiale (p150) zeigen erhöhte Amplituden sowie erniedrigte Latenzen, d.h. die Aktivität tritt früher und stärker auf. Dies kann mit der für die schwierigere Aufgabe notwendigerweise verstärkten Aufmerksamkeitsleistung erklärt werden. Dagegen treten die später evozierten Potentiale (p300) bei erhöhter Anforderung mit verlängerter Latenz und niedrigerer Amplitude auf. Zudem bewegen sich die Feldschwerpunkte auseinander. Dies kann zwar auf eine diffusere und oberflächlichere Aktivierung von Neuronenverbänden aufgrund der komplexeren Aufgabenstellung hinweisen. Da jedoch auch die Fehlerrate anstieg, ist als plausible Erklärung anzunehmen, dass die synchrone Neuronenaktivität, die zur Aufgabenbewältigung nötig ist, bei Überschreiten der Leistungsgrenze Unregelmäßigkeiten im Sinne fehlerhafter Prozesse aufweist, die sich im Mittelwert der hirnelektrischen Potentiale als Amplitudenreduktion, auf Verhaltensebene durch falsche Antworten abbilden.

Die Daten der Verhaltensebene stimmen mit den Erkenntnissen diverser neuropsychologischer Untersuchungen überein. Die Ergebnisse aus den kognitiv-evozierten Potentialen passen gut zu den aus der Literatur bekannten modulierenden Variablen Komplexität und Aufmerksamkeit.

In dieser Studie konnte nun erstmals mit einer hochauflösenden EEG-Untersuchung systematisch der modulierende Einfluss des Schwierigkeitsgrades des SAT auf die Komponenten der evozierten Potentiale nachgewiesen werden, d.h. inwieweit Komponenten der evozierten Potentiale graduell je nach Schweregrad der Aufgabe

verändert werden. Gerade in der Forschung mit schizophren erkrankten Patienten wurde der SAT als Verhaltensmaß eingesetzt. Diese Patientengruppe zeichnet sich durch Defizite im Bereich der Aufmerksamkeitsleistung und folglich Defizite in der Bearbeitung komplexerer Aufgaben aus. Aus der Kombination beider Methoden lassen sich in Zukunft bei dem auch für diese Patientengruppe leicht durchzuführenden Test Aussagen über die Veränderung der hirnelektrischen Aktivität gegenüber gesunden Probanden erwarten.

Als wesentliche Schlussfolgerung für die Aktivierung von messbaren hirnphysiologischen Prozessen durch kognitive Aufgaben kann festgestellt werden, dass eine Steigerung des Schwierigkeitsgrades über die optimale individuelle Leistung hinaus zur Kontamination der Messungen mit fehlerhaften Prozessen zu führen scheint und somit zu schwer oder nicht interpretierbaren Ergebnissen führt. Dies erfordert ein grundsätzliches Umdenken bei der Anwendung und Entwicklung von neurophysiologischen Tests in der modernen funktionellen Hirnforschung, da diese Tests traditionell v. a. durch Variablen auf Verhaltensebene beurteilt werden (nämlich Fehlerraten), die erst dann aussagekräftig werden, wenn das System hirnphysiologisch gesehen bereits übersteuert ist.

6 Literaturverzeichnis

1. Asarnow R-F, MacCrimmon D-J: Residual performance deficit in clinically remitted schizophrenics: A marker of schizophrenia? Journal-of-Abnormal-Psychology 1978; 87:597-608

2. Asarnow RF, MacCrimmon DJ: Span of apprehension deficits during the postpsychotic stages of schizophrenia. A replication and extension. Arch.Gen.Psychiatry 1981; 38:1006-1011

3. Asarnow RF, MacCrimmon DJ: Attention/information processing, neuropsychological functioning, and thought disorder during the acute and partial recovery phases of schizophrenia: a longitudinal study. Psychiatry Res. 1982; 7:309-319

4. Asarnow RF, Nuechterlein KH, Marder SR: Span of apprehension performance, neuropsychological functioning, and indices of psychosis-proneness. J.Nerv.Ment.Dis. 1983; 171:662-669

5. Benedict RH, Harris AE, Markow T, McCormick JA, Nuechterlein KH, Asarnow RF: Effects of attention training on information processing in schizophrenia. Schizophr.Bull. 1994; 20:537-546

6. Bowen L, Wallace CJ, Glynn SM, Nuechterlein KH, Lutzker JR, Kuehnel TG: Schizophrenic individuals' cognitive functioning and performance in

interpersonal interactions and skills training procedures. J.Psychiatr.Res. 1994; 28:289-301

7. Buchanan RW, Strauss ME, Breier A, Kirkpatrick B, Carpenter WT, Jr.: Attentional impairments in deficit and nondeficit forms of schizophrenia. Am.J.Psychiatry 1997; 154:363-370

8. Caplan R, Foy JG, Asarnow RF, Sherman T: Information processing deficits of schizophrenic children with formal thought disorder. Psychiatry Res. 1990; 31:169-177

9. Davidson GS, Neale JM: The effect of signal-noise similarity on visual information processing of schizophrenics. Journal of Abnormal Psychology 1974; 83:683-686

10. Estes W-K, Taylor H-A: A technique for assessing variability of perceptual span. Proceedings of the National Acadamy of Sciences 1965; 54:403-407

11. Fallgatter AJ, Brandeis D, Strik WK: A robust assessment of the NoGo-anteriorisation of P300 microstates in a cued Continuous Performance Test. Brain Topography 1997; 9:295-302

12. Fallgatter AJ, Mueller TJ, Strik WK: Neurophysiological correlates of mental imagery in different sensory modalities. International Journal of Psychophysiology 1997;145-153

13. Fallgatter AJ, Strik WK: The NoGo-anteriorization as a neurophysiological standard-index for cognitive response control. International Journal of Psychophysiology 1999; 32:233-238

14. Gevins AS, Smith ME: Neurophysiological measures of working memory and individual differences in cognitive ability and cognitive style. Cerebral Cortex 2000; 10:829-839

15. Gomez CM, Vazquez M, Vaquero E, Lopez-Mendoza D, Cardoso MJ: Frequency analysis of the EEG during spatial selective attention. International Journal of Neuroscience 1998; 95:17-32

16. Grune K, Metz AM, Hagendorf H, Fischer S: Information processing in working memory and event-related brain potentials. International Journal of Psychophysiology 1996; 23:111-120

17. Hain C, Maier W, Klingler T, Franke P: Positive/negative symptomatology and experimental measures of attention in schizophrenic patients. Psychopathology 1993; 26:62-68

18. Harris A, Ayers T, Leek MR: Auditory span of apprehension deficits in schizophrenia. J.Nerv.Ment.Dis. 1985; 173:650-657

19. Ito M, Kanno M, Mori Y, Niwa S: Attention deficits assessed by Continuous Performance Test and Span of Apprehension Test in Japanese schizophrenic patients. Schizophr.Res. 1997; 23:205-211

20. Kotchoubey B, Wascher E, Verleger R: Shifting attention between global features and small details: an event-related potential study. Biological Psychology 1997; 46:25-50

21. Lehmann D: Multichannel topography of human alpha EEG fields. Electroencephalography Clinical Neurophysiology 1971; 31:439-449

22. Lehmann D: Spatial analysis of EEG and evoked potential data. Duffy FH (ed.s) 1986;

23. Lehmann D: Principles of spatial analysis. in Handbook of electroencephalography and clinical neurophysiology, vol. 1. Edited by Gevins AS, Remond A. Amsterdam, Elsevier, 1987

24. Lehmann D, Skrandies W: Reference-free identification of components of checkerboard-evoked multichannel potential fields. Electroencephalography Clinical Neurophysiology 1980; 48:609-621

25. Lehmann D, Skrandies W: Spatial analysis of evoked potentials in man: an overview. Progr.Neurobiol. 1984; 23:227-250

26. Maier W, Franke P, Hain C, Kopp B, Rist F: Neuropsychological indicators of the vulnerability to schizophrenia. Prog.Neuropsychopharmacol.Biol.Psychiatry 1992; 16:703-715

27. Maurer K, Lowitzsch K, Stöhr M: AEP-VEP-SEP; Atlas mit Einführungen, 2 ed. Stuttgart, Enke, 1990

28. McEvoy LK, Smith ME, Gevins AS: Dynamic cortical networks of verbal and spatial working memory: effects of memory load and task practice. Cerebral Cortex 1998; 8:563-574

29. Mueller TJ, Federspiel A, Fallgatter AJ, Strik WK: EEG signs of vigilance fluctuations preceding perceptual flips in multistable illusionary motion. Neuroreport 1999; 10:

30. Neale JM, McIntyre CW, Fox R, Cromwell RL: Span of apprehension in acute schizophrenics. J.Abnorm.Psychol. 1969; 74:593-596

31. Pascual-Marqui RD, Michel CM, Lehmann D: Low resolution electromagnetic tomography: a new method for localizing electrical activity in the brain. Int J Psychophysiol 1994; 18:49-65

32. Rund BR, Orbeck AL, Landro NI: Vigilance deficits in schizophrenics and affectively disturbed patients. Acta Psychiatr.Scand. 1992; 86:207-212

33. Städtler T: Psychodiagnostik in Lexikon der Psychologie Alfred Kröner Verlag, 1998

34. Strandburg RJ, Marsh JT, Brown WS, Asarnow RF, Guthrie D: Event-related potential concomitants of information processing dysfunction in schizophrenic children. Electroencephalogr.Clin.Neurophysiol. 1984; 57:236-253

35. Strandburg RJ, Marsh JT, Brown WS, Asarnow RF, Guthrie D: Information-processing deficits across childhood- and adult- onset schizophrenia [published erratum appears in Schizophr Bull 1995;21(2):251]. Schizophr.Bull. 1994; 20:685-695

36. Strandburg RJ, Marsh JT, Brown WS, Asarnow RF, Guthrie D, Higa J: Reduced attention-related negative potentials in schizophrenic children. Electroencephalogr.Clin.Neurophysiol. 1991; 79:291-307

37. Strandburg RJ, Marsh JT, Brown WS, Asarnow RF, Guthrie D, Higa J: Event-related potentials in high-functioning adult autistics: linguistic and nonlinguistic visual information processing tasks. Neuropsychologia. 1993; 31:413-434

38. Strandburg RJ, Marsh JT, Brown WS, Asarnow RF, Guthrie D, Higa J, Yee Bradbury CM, Nuechterlein KH: Reduced attention-related negative potentials in schizophrenic adults. Psychophysiology 1994; 31:272-281

39. Strauss ME, Prescott CA, Gutterman DF, Tune LE: Span of apprehension deficits in schizophrenia and mania. Schizophr.Bull. 1987; 13:699-704

40. Strik, W. K. Mikrozustände der hirnelektrischen Felder: Methode und Ergebnisse bei Gesunden und psychiatrischen Patienten. 25. 1993. Psychiatrische Klinik und Poliklinik der Universität Würzburg.
Ref Type: Thesis/Dissertation

41. Strik WK, Dierks T, Franzek E, Stober G, Maurer K: P300 in schizophrenia: interactions between amplitudes and topography. Biol Psychiatry 1994;850-856

42. Strik WK, Lehmann D: Data-determined window size and space-orientated segmentation of spontaneous EEG map series. Electroencephalography Clinical Neurophysiology 1993; 87:169-174

43. Tarnowski KJ, Prinz RJ, Nay SM: Comparartive analysis of attentional deficits in hyperactive and learning-disabled children. Psychiatry-Research 1986; 95:341-345

44. Voruganti LN, Heslegrave RJ, Awad AG: Neurocognitive correlates of positive and negative syndromes in schizophrenia. Can.J.Psychiatry 1997; 42:1066-1071

45. Warrington EK, Kinsbourne M, James M: Uncertainty and transitional probability in the span of apprehension. Br.J.Psychol. 1966; 57:7-16

46. Wright MJ, Geffen GM, Geffen LB: Event related potentials during covert orientation of visual attention: effects of validity and directionality. Biological Psychology 1995; 41:183-202

7 Anhang

7.1.1 Gleichungen

Average Reference. u ist gleich der referenzunabhängigen Amplitude, v ist die gemessene Voltage, n die Anzahl der Elektroden.

Gleichung 1: Average Reference

$$u = v - \frac{\sum_{i=1}^{n} v_i}{n}$$

Globale Feldstärke. n ist gleich der Anzahl der Elektroden, u ist gleich der gemessenen Voltage an der Elektrode.

Gleichung 2: Globale Feldstärke

$$\sqrt{\frac{1}{2n} \sum_{i=1}^{n} \sum_{j=1}^{n} \left(u_i - u_j\right)^2}$$

Centroid-Lokalisation. Die positiven und negativen Areale der Karte, sowie die Lokalisationen für die links-rechts (x) und für die antero-posteriore (y) Koordinate werden getrennt berechnet. Voraussetzung ist die average reference. v ist gleich der Amplitude (Amplituden > 0 für die Berechnung des positiven, < 0 für die Berechnung des negativen Centroiden), l gleich der Koordinate (x oder y) der Elektrode, n ist die Anzahl der Elektroden.

Gleichung 3: Centroid-Lokalisation

$$\frac{\sum_{i=1}^{n} v_i l_i}{\sum_{i=1}^{n} v_i}$$

7.1.2 Anamnesebogen

Anamnesebogen SAT - Studie

evozierte Potentiale beim Span-of-Apprehension-Test (**SAT)**
mit 64-Kanal-EEG-Aufzeichnung

Probandennummer: Untersuchungsdatum:

Name: Vorname:

Geburtsdatum: Geschlecht:

Schulbildung:

Beginn: Ende:

Händigkeit:	
Visus:	
Medikamente:	
Psychiatrische Vorerkrankungen:	
Familienanamnese:	
Sonstige Besonderheiten:	

8 Quellcodes in C und Textdateiinhalte

8.1 SAT.C

Es folgt der Quellcode des Span-Of-Apprehension-Test-Simulationsprogramms,

das von S. Dannemann erstellt wurde.

```
//cl /AL /c sat.c
//link sat.obj mouse.lib tmr.lib
//c: sat
#include <malloc.h>
#include <graph.h>     //für fonts + gitter
#include <time.h>
#include <stdio.h>     //für printf
#include <math.h>
#include <conio.h>     //für getch()
#include <string.h>
#include <stdlib.h>
#include <io.h>        //für open
#include <fcntl.h>     //für open
#include "mouse.h"
#include "tmr.h"

#define STIM_ADDRESS          888
#define HOLD                    0
#define COMMENT                 1
#define MOUSE                   1
#define ESC               27
#define FGC                _BLACK
#define BGC                    _BRIGHTWHITE
#define BUTTONANZ              1
#define YEXT                    0         //vertikale
Buchstabenkorrektur
#define RES                "vga"
#define FELDER                  4
#define BREITE                100
#define F                  70         //Groß: 70, Klein: 102
#define SCHRIFTGROESSE           "100"
#define SCHRIFTART               "*.fon"
#define MAXTIME     2147483647
#define SECTIME        1000000
#define EXPOSURETIME           500
#define WARTEN                3000
#define MAXWAIT         2000
#define M_RESET           16
#define MAXTRIALS        500
#define MAXLETTER         16
#define MAXZEILEN         33
#define getrandom( min, max ) ((rand() % (int)(((max)+1) - (min))) +
(min))

const int
  maxtrials = MAXTRIALS,
  maxletter = FELDER * FELDER;
long
  exposuretime = EXPOSURETIME,
  warten = WARTEN,
  maxwait = MAXWAIT,
```

```
  start,
  trial_start[MAXTRIALS],
  reaktion[MAXTRIALS],
  fgcol = FGC,
  bgcol = BGC;
double
  richtige = 0;
  //richtigeletter[17];
int
  //trialsletter[17],
  lanz[MAXTRIALS],                  // Anzahl Buchstaben im aktuellen
Trial
  letterX[MAXLETTER],
  letterY[MAXLETTER],
  feldbreite = BREITE,
  winX1, winY1, winX2, winY2,
  richtig[MAXTRIALS],
  Fdrin[MAXTRIALS],
  Yextent = YEXT,
  hold = HOLD,
  comment = COMMENT,
      topo = MOUSE,
      buttonanz = BUTTONANZ,
  address = STIM_ADDRESS;
  stop = 0,
  back = 1,
  script = 2,
  gitter = 3,
  markeranz = 0,
  marker_letter[MAXLETTER];
short
  portv;
char
  *trial[MAXTRIALS],
  *schriftart,
  *schriftgroesse = SCHRIFTGROESSE,
  font[60+1]="bh",       //   "t'roman'h100w100",
  vres[12+1] = RES;
struct videoconfig vc;
struct _fontinfo fi;

void ini(void);
char* setpointer(size_t laenge);
FILE* setFILEpointer(size_t laenge);
int buchstabenlesen(char *dateiname);
void buchstabenausgabe(int trialnr);
int output(int trialnr);
void ergebnis(char *trials_dateiname, char *dateiname, int trialanz);
void syntax(void);
void init_font(void);
void init_graph(void);
void gitter2(void);
void close_graph(void);
void gitter_control(void);
void abbruch(char *t);
int mouse_wait(int trialnr, long delai);
void set_tmr(void);
long get_time(void);
long timer(void);
void tmr_warten(long delai);
long c_output(char *zeichen, long color);

void init_stim_port(void) { outp(1823,0x88);         }
```

```
void port(int val) { outp(address, val); }
void gk_sync(void) { while(!(inp(0x3DA) & 8)); } // if bit #3 is set,
vertical retrace is
                                                    // done;

void tmr_wait(long,long);
void delay(long);
void resp_wait(void);
void marker_wait(void);


void main(int argc, char *argv[]) {
  int trialanz, trialnr;

  if (argc == 3) {
    if(address == STIM_ADDRESS) init_stim_port();
    ini();
    init_font();
    init_graph();
    mouse_init();
    set_tmr();
    c_output("Tastendruck startet SAT", (long)BGC);
           port(comment*2-1);
    //printf("%i\n", comment);
    getch();
           port(hold);
    //c_output("Tastendruck startet SAT", (long)FGC);
    gitter_control();
    trialanz = buchstabenlesen(argv[1]);
    for (trialnr = 0; trialnr < trialanz; trialnr++) {
      tmr_warten(warten);
      buchstabenausgabe(trialnr);
    }
    ergebnis(argv[1], argv[2], trialanz);
    free(schriftart);
    for (trialnr = 0; trialnr < maxtrials; trialnr++)
      free(trial[trialnr]);
    //getch();
    abbruch("Das SAT-Programm ist beendet\n\r");
  }
  else syntax();
}

//in Bildschirmmitte zentrieren
long c_output(char *zeichen, long color) {

  _moveto(vc.numxpixels / 2 - _getgtextextent(zeichen) / 2,
        vc.numypixels / 2 - atoi(schriftgroesse) / 2);
  _remappalette(4, (long)FGC);
  _setcolor(4);
  _outgtext(zeichen);
  gk_sync();
  _remappalette(4, color);
  return(timer());
}

void ini() {
  FILE *datei;
  char *line, *puffer;
      int counter = 0, distractor;

  line = setpointer(80);
  puffer = setpointer(80);
```

```
datei = setFILEpointer(sizeof(datei));
schriftart = setpointer(15);
schriftart = SCHRIFTART;
for(counter = 0; counter < maxtrials; counter++) {
  lanz[counter] = 0;
  trial_start[counter] = 0;
  reaktion[counter] = 0;
  Fdrin[counter] = 0;
  richtig[counter] = 0;
}
for(counter = 0; counter < maxletter; counter++) {
  letterX[counter] = 0;
          letterY[counter] = 0;
          marker_letter[counter] = -2;
    }
    if ( (datei = fopen("sat.ini", "rt") ) != NULL) {
    fgets(line, 80, datei);
    fgets(line, 80, datei);
  fgcol = atol(line);
  if (fgcol == 1) fgcol = BGC;
    fgets(line, 80, datei);
    fgets(line, 80, datei);
  bgcol = atol(line);
          if (bgcol == 1) bgcol = BGC;

                fgets(line, 80, datei);
                fgets(line, 80, datei);
          buttonanz = atoi(line);

    fgets(line, 80, datei);
    fgets(line, 80, datei);
  exposuretime = atol(line);
    fgets(line, 80, datei);
    fgets(line, 80, datei);
  warten = atol(line);
    fgets(line, 80, datei);
    fgets(line, 80, datei);
  maxwait = atol(line);
    fgets(line, 80, datei);
    fgets(line, 80, datei);
  strncpy(vres, line, strlen(line)-1);
    fgets(line, 80, datei);
    fgets(line, 80, datei);
  strncpy(schriftart, line, strlen(line)-1); // /n herausschneiden
    fgets(line, 80, datei);
    fgets(line, 80, datei);
  strncpy(schriftgroesse, line, strlen(line)-1);
    fgets(line, 80, datei);
    fgets(line, 80, datei);
  Yextent = atoi(line);
    fgets(line, 80, datei);
    fgets(line, 80, datei);
  feldbreite = atoi(line);
    fgets(line, 80, datei);
    fgets(line, 80, datei);
  hold = atoi(line);
    fgets(line, 80, datei);
    fgets(line, 80, datei);
  address = atoi(line);
    fgets(line, 80, datei);
    fgets(line, 80, datei);
    fgets(line, 80, datei);
    do {
```

```
      fgets(line, 80, datei);
      if(atoi(line) != -1) {
        distractor = atoi(line);
        fgets(line, 80, datei);
        marker_letter[distractor] = atoi(line);
        markeranz++;
      }
      } while(atoi(line) != -1);
  }
  else {
    //fclose(datei);  bei else ist die Datei doch gar nicht geoeffnet
!
    datei = fopen("sat.ini", "wt");

    fprintf(datei, "Schriftfarbe\n");
    if (fgcol == BGC) fprintf(datei, "%d\n", 1);
      else fprintf(datei, "%ld\n", fgcol);
    fprintf(datei, "Hintergrundfarbe\n");
    if (bgcol == BGC) fprintf(datei, "%d\n", 1);
      else fprintf(datei, "%ld\n", bgcol);

            fprintf(datei, "Mausknopf-Anzahl ( 1 oder 2 )\n");
            fprintf(datei, "%d\n", buttonanz);
    fprintf(datei, "Expositionszeit in ms\n");
    fprintf(datei, "%ld\n", exposuretime);
    fprintf(datei, "Warten nach Tastendruck in ms\n");
    fprintf(datei, "%ld\n", warten);
            fprintf(datei, "maximales Warten auf Tastendruck in ms\n");
    fprintf(datei, "%ld\n", maxwait);
    fprintf(datei, "Video-Auflösung\n");
    fprintf(datei, "%s\n", vres);
    fprintf(datei, "Schriftart\n");
    fprintf(datei, "%s\n", schriftart);
    fprintf(datei, "Schriftgröße\n");
    fprintf(datei, "%s\n", schriftgroesse);
    fprintf(datei, "Höhenberechnung (0 = pro Letter, 1 = Höhe, 2 - . =
individuell)\n");
    fprintf(datei, "%d\n", Yextent);
    fprintf(datei, "Feldbreite in Pixel\n");
    fprintf(datei, "%d\n", feldbreite);
    fprintf(datei, "Hold-Value\n");
    fprintf(datei, "%d\n", hold);
    fprintf(datei, "Port\n");
    fprintf(datei, "%d\n", address);
    fprintf(datei, "1.Zeile: Distractorenanzahl\n");
    fprintf(datei, "2.Zeile: Marker für 'F' (folgenden Marker für 'T'
frei lassen)\n");
    fprintf(datei, "letzte Zeile: -1\n");
    fprintf(datei, "-1\n");
  }
  fclose(datei);
  free(line);
  free(puffer);
  free(datei);
}

FILE* setFILEpointer(size_t laenge) {
  FILE *pointer;

  if ((pointer = (FILE *) malloc(laenge)) == NULL)
    abbruch("nicht genug Speicher, um buffer zu allocaten\n");
    // Programm beenden, wenn nicht genug Speicher
  return(pointer);
```

```
}

char* setpointer(size_t laenge) {
  char *pointer;

  if ((pointer = (char *) malloc(laenge)) == NULL)
    abbruch("nicht genug Speicher, um buffer zu allocaten\n");
    // Programm beenden, wenn nicht genug Speicher
  return(pointer);
}

int buchstabenlesen(char *dateiname) {
  FILE *datei;
  int  nr, trialnr;
  char letter = ' ', *line;

  datei = setFILEpointer(sizeof(datei));
  if (( datei = fopen(dateiname, "rt") ) == NULL) {
    //clrscr();
    fprintf(stderr, "Cannot open input file.\n");
    getch();
    free(datei);
    exit(1);
  }

  for(trialnr = 0; trialnr < maxtrials; trialnr++)
    trial[trialnr] = setpointer(maxletter);

  while (letter != '\n') letter = fgetc(datei); //ausblenden von
erster Zeile
  trialnr = 0;
  do {
    fgets(line, 80, datei);
    lanz[trialnr] = atoi(line);
    nr = 0;
    do {
      letter = fgetc(datei);
      if (letter != EOF) {
      sprintf(trial[trialnr], "%c", letter);
      trial[trialnr]++;
      nr++;
      }
    } while ( (letter != '\0') && (letter != EOF) );
    if (letter != EOF) {
      letter = fgetc(datei); //Absaegen von '\n'
      trial[trialnr] -= nr;
      trialnr++;
    }
  } while (letter != EOF);
  fclose(datei);
  free(datei);
  return(trialnr);
}

void buchstabenausgabe(int trialnr) {           //druckt die
Buchstaben
 int stimulus = 0, m, distractoranz; //wenn "F" enthalten, stimulus =
1
 int bedingung, bedingung2; //um 1BR oder 2BR zu ermoeglichen

  _setcolor(script);
  stimulus = output(trialnr);
  gk_sync();
```

```
  _remappalette(script, fgcol);
  distractoranz = lanz[trialnr] - 1;
  if(stimulus == 1) {
                port((hold + marker_letter[distractoranz])*2-1);
                //printf("%i\n", hold +
marker_letter[distractoranz]);
    }
    else {
                port((hold + marker_letter[distractoranz] + 1)*2-1);
                //printf("%i\n", hold + marker_letter[distractoranz]
+ 1);
    }
  tmr_warten(M_RESET);
      port(hold);
  trial_start[trialnr] = timer(); //Zeitpunkt des jeweiligen
Trialstarts
  m = mouse_wait(trialnr, exposuretime); //Exposuretime -in Bezug auf
den ersten
                                    //Trialstart
  gk_sync();
  _remappalette(script, bgcol);
  _setcolor(back);
      output(trialnr);
      if (m == 0) m = mouse_wait(trialnr, maxwait);
      //    else mouse_wait(-1, maxwait);

      if (buttonanz == 1) {
            if ( (m == 1) || (m == 2) ) bedingung = 1;
                  else bedingung = 0;
            if (m == 0) bedingung2 = 1;
                  else bedingung2 = 0;
      }
      if (buttonanz == 2) {
            if (m == 1) bedingung = 1;
                  else bedingung = 0;
            if (m == 2) bedingung2 = 1;
                  else bedingung2 = 0;
      }

      if (bedingung == 1) {
            port((topo + 1)*2-1);
    //printf("%i\n", topo + 1);
    tmr_warten(M_RESET);
            port(hold);
    if (stimulus) {
      richtige++;
      Fdrin[trialnr] = 1;
      richtig[trialnr] = 1;
    }
    else {
      Fdrin[trialnr] = 0;
      richtig[trialnr] = 0;
    }
  }
      if (bedingung2 == 1) {
            port((topo + 2)*2-1);
    //printf("%i\n", topo + 2);
    tmr_warten(M_RESET);
            port(hold);
    if (stimulus) {
      Fdrin[trialnr] = 1;
      richtig[trialnr] = 0;
    }
```

```
    else {
      richtige++;
      Fdrin[trialnr] = 0;
      richtig[trialnr] = 1;
    }
  }
}

int output (int trialnr) {
 char *ausgabe, *trialakut;
 int laenge, letterNr = 0, pos = 0, stimulus = 0;

  ausgabe = setpointer(1);
  trialakut = setpointer(16);
  trialakut = trial[trialnr];
  laenge = strlen(trialakut);
  for(pos = 0; pos < laenge; pos++) {
    ausgabe = "";
    strncpy(ausgabe, trialakut, 1);
    ausgabe = strrev(ausgabe) + strlen(ausgabe) - 1;
    trialakut++;
    if (Yextent == 0) Yextent = fi.pixheight;
    if (Yextent == 1) Yextent = fi.ascent;
    _moveto(letterX[letterNr] - _getgtextextent(ausgabe) / 2,
          letterY[letterNr] - Yextent / 2);
    letterNr++;
    if(*ausgabe == F) stimulus = 1;
           _outgtext(ausgabe);
      }
  return(stimulus);
  free(trialakut);
  free(ausgabe);
}

void ergebnis (char *trials_dateiname, char *dateiname, int trialanz)
{

  int
    counter1,
    counter2,
    counter3,
    trials_pro_letter[17];
  double
    reaktion_pro_letter[17],
    reaktion_total = 0,
    richtige_pro_letter[17],
    quotient;
  FILE
    *datei;

  counter1 = 0;
  for (counter1 = 0; counter1 < 17; counter1++) {
    trials_pro_letter[counter1] = 0;
    richtige_pro_letter[counter1] = 0;
    reaktion_pro_letter[counter1] = 0;
  }
  datei = setFILEpointer(sizeof(datei));
  datei = fopen (dateiname, "wt");
  // printf("Die Ergebnisse werden gespeichert");
  fprintf(datei, "%s\n", trials_dateiname);
  counter2 = 0;
  for (counter2 = 0; counter2 < trialanz; counter2++) {
    trials_pro_letter[lanz[counter2]]++;
```

```
    reaktion_pro_letter[lanz[counter2]] += reaktion[counter2] -
trial_start[counter2];
    if(richtig[counter2]) richtige_pro_letter[lanz[counter2]]++;
    reaktion_total += reaktion[counter2] - trial_start[counter2];
    fprintf(datei, "%d\t%d\t%.0ld\t%.0ld\t%.0ld\t%d\t%d\n",
      counter2 + 1, lanz[counter2], trial_start[counter2] / 1000,
reaktion[counter2] / 1000,
      (reaktion[counter2] - trial_start[counter2]) / 1000,
      Fdrin[counter2], richtig[counter2]);
  }
  quotient = (richtige / trialanz);
  fprintf(datei, "%.0f\t/\t%d\t=\t%.2f\t%.0f\n", richtige, trialanz,
quotient, reaktion_total / trialanz / 1000);
  counter3 = 1;
  for (counter3 = 1; counter3 < 17; counter3++)
    if(trials_pro_letter[counter3] > 0)
      fprintf(datei, "%d\t%.0f\t/\t%d\t=\t%.2f\t%.0f\n",
      counter3, richtige_pro_letter[counter3],
trials_pro_letter[counter3],
      (richtige_pro_letter[counter3] / trials_pro_letter[counter3]),
      (reaktion_pro_letter[counter3] / trials_pro_letter[counter3] /
1000));
  fclose(datei);
  free(datei);
}

void syntax() {
  printf("\n\rSAT (Span of Apprehension Task) - Programm\n\r");
      printf("Version 3.1, (C) August 1997, Sven Dannemann\n\n\r");
  printf("  Syntax: sat <trialsdateiname> <ergebnisdateiname>\n\r");
  printf("Beispiel: sat trials.dat ergebnis.log\n\r");
}

void init_graph(void) {
    switch(vres[0]) {
      case 's':
      case 'S': if(!_setvideomode(_XRES16COLOR)) abbruch("video mode
not supported"); break;
      case 'v':
      case 'V': if(!_setvideomode(_VRES16COLOR)) abbruch("video mode
not supported"); break;
      case 'e':
      case 'E': if(!_setvideomode(_ERESCOLOR)) abbruch("video mode not
supported"); break;
      case 'c':
      case 'C': if(!_setvideomode(_MRES16COLOR)) abbruch("video mode
not supported"); break;
    }
    _getvideoconfig(&vc);
    if (vc.numvideopages < 1) abbruch("minimum 2 video pages needed");
    _remappalette(back, bgcol);
    _remappalette(gitter, bgcol);
    _remappalette(script, bgcol);
}

void close_graph(void) {
  _setvideomode(_TEXTC80);
  _clearscreen(_GCLEARSCREEN);
}

void init_font(void) {
  if(_registerfonts(schriftart) < 0 ) abbruch("no font in working
directory");
```

```
  strcat(font, schriftgroesse);
  _setfont(font);
  _getfontinfo(&fi);
}

void gitter2() {
//zeichnet ein Gitter mit felder*felder Kästchen der Pixelbreite
feldbreite

  //felder ist die Anzahl der Kästchen einer Seite
  //feldbreite ist die Pixelbreite eines Kästchens
  int horiz, vertikal, felder = FELDER, zahler = 0,
      winX1 = (vc.numxpixels - (felder * feldbreite)) / 2,
      winY1 = (vc.numypixels - (felder * feldbreite)) / 2,
      winX2 = winX1 + (felder * feldbreite),
      winY2 = winY1 + (felder * feldbreite);

  //Hintergrund weiß machen, zentriertes Fenster öffnen
  _setcolor(back);
  _rectangle(_GFILLINTERIOR,0,0,vc.numxpixels,vc.numypixels);
  _setviewport(winX1, winY1, winX2, winY2);

  _setcolor(gitter);
  //zeichnen der horizontalen Linien
  for (horiz = 0; horiz <= felder; horiz++) {
    _moveto(0, horiz * feldbreite);
    _lineto(felder * feldbreite, horiz * feldbreite);
    //festlegen der Buchstaben-X-Koordinaten
    if (horiz < felder) //5 Linien gibt 4 Kästchen, also 4 Buchstaben
      for (zahler = horiz; zahler < maxletter; zahler += felder)
      letterX[zahler] = feldbreite * (0.5 + horiz);
  }

  //zeichnen der vertikalen Linien
  for (vertikal = 0; vertikal <= felder; vertikal++) {
    _moveto(vertikal * feldbreite, 0);
    _lineto(vertikal * feldbreite, felder * feldbreite);
    //festlegen der Buchstaben-Y-Koordinaten
    if (vertikal < felder)
      for (zahler = vertikal * felder; zahler < maxletter; zahler++)
      letterY[zahler] = feldbreite * (0.5 + vertikal);
  }
}

void gitter_control() {
  gitter2();
  _remappalette(gitter, fgcol);
}

int mouse_wait(int trialnr, long delai) {
  int  m = 3;
  long zeit;

  zeit = timer();
  while ( ( !(m = mouse_state()) || ((m != 1) && (m != 2)) )
       && ( timer() < zeit + delai * 1000 ) )
    if(kbhit()) if(getch() == ESC) abbruch("");
  if (trialnr != -1) reaktion[trialnr] = timer();
  return(m);
}

void set_tmr(void) {
    //start = tmr_clock((int)_CLK_RUN);
```

```
    start = tmr_clock(_CLK_RUN,(long)MAXTIME,&stop);
}

long get_time() {
  static long mem = 0;

  if(stop || (tmr_clock(_CLK_TIME) > (long)MAXTIME - (long)SECTIME)) {
    mem += tmr_clock(_CLK_STOP);
    set_tmr();
    printf("\a");
  }
  return (tmr_clock(_CLK_TIME) + mem);
}

long timer() {
  long zeitpunkt;
  long time = 0;

  time = get_time();    // = tmr_clock((int)_CLK_TIME);
  zeitpunkt = (time - start);
  return(zeitpunkt);
}

void tmr_warten(long delai) {
    long time = 0;

    time = get_time();  // = tmr_clock((int)_CLK_TIME);
    while ( get_time() < (time + (delai*1000)))
      if(kbhit()) if(getch() == ESC) abbruch("");
}

void abbruch(char *t) {
  close_graph();
  printf("\n%s",t);
  exit(0);
}
```

8.2 SAT100.INI

Es folgt der Ausdruck der für das SAT-Programm benötigten Initialsierungsdatei (Textdatei).

Sie stellt das SAT-Programm in SAT1-Modus ein.

```
Schriftfarbe
0
Hintergrundfarbe
1
Mausknopf-Anzahl ( 1 oder 2 )
2
Expositionszeit in ms
100
Warten nach Tastendruck in ms
2000
maximales Warten auf Tastendruck in ms
2000
Video-Aufloesung
vga
Schriftart
*.fon
Schriftgröße
100
Höhenberechnung (0 = pro Letter, 1 = Höhe, 2 - . = individuell)
0
Feldbreite in Pixel
100
Hold-Value
0
Port
888
1.Zeile: Distraktorenanzahl
2.Zeile: Marker für 'F' (folgenden Markerwert für 'T' frei lassen)
letzte Zeile: -1
0
4
2
6
4
8
9
10
-1
```

8.3 TRIALS.C

Es folgt der Quellcode des Trials-Programms,

das die Zeichenketten für das SAT-Programm erstellt

und sie in der Datei Trials.dat speichert.

```
// cl /AL /F FFF trials.c      so ist stack auf 4k
#include <graph.h>                          //für clearscreen
#include <stdio.h>
#include <io.h>                                   //für eof
#include <stdlib.h>                         //für random
#include <conio.h>
#include <string.h>
#include <time.h>                           //für randomize

#define MAX_TRIALS    500
#define LETTER_ANZAHL   8
#define ALPHABET       65               //65 für große Buchstaben, 97
für kl.
#define F              ALPHABET + 5
#define T              ALPHABET + 19
#define getrandom( min, max ) ((rand() % (int)(((max)+1) - (min))) +
(min))

void maketrials(FILE *datei, const maxtrials, int lanz[], int
ftotal[]);//lanz = Buchstaben
int zufallrechnung(int von, int bis, int randzahl[]);
int set_f_oder_t(int f[], int maxtrials);
void set_letters(FILE *datei, int lanz, int f);         //pro trial
int set_letter_positions(int pos[], int lanz);
void syntax();

void main(int argc, char *argv[]) {
  FILE *datei;       //Programmaufruf:
  int nr,       //Dateiname Trialzahl Buchstabenanzahl Trialanzahl Bu..
      nr2,
      trialanz = 0,
      f[MAX_TRIALS],
      ftotal[MAX_TRIALS],
      lanz[MAX_TRIALS];

  if (argc > 3) {
    datei = fopen(argv[1], "wt");
    for (nr = 2; nr < argc; nr += 2)
      fprintf(datei, "%s %s ", argv[nr], argv[nr+1]); //1.Zeile
    fprintf(datei, "\n");

    for (nr = 2; nr < argc; nr += 2) {
      *f = set_f_oder_t(f, atoi(argv[nr]));
      for (nr2 = 0; nr2 < atoi(argv[nr]); nr2++) {
      ftotal[trialanz] = f[nr2];
      lanz[trialanz] = atoi(argv[nr+1]);
      trialanz++;
      }
    }
    maketrials(datei, trialanz, lanz, ftotal);
    fclose(datei);
  }
  else syntax();
```

```
}

void maketrials(FILE *datei, const maxtrials, int lanz[], int fto-
tal[]) {
  int trialnr,
      gemischt = 1,
      randnr[MAX_TRIALS];

  _clearscreen(_GCLEARSCREEN);
  if (gemischt == 1)
    *randnr = zufallrechnung(0, maxtrials, randnr);
  for (trialnr = 0; trialnr < maxtrials; trialnr++) {
    if (gemischt != 1) randnr[trialnr] = trialnr;
    fprintf(datei, "%d\n", lanz[randnr[trialnr]]);    //vor jedem
trial
    set_letters(datei, lanz[randnr[trialnr]],
ftotal[randnr[trialnr]]);
    fputc('\0', datei);
    fputc('\n', datei);
    printf("\n\r");
  }
  getch();
}

int zufallrechnung(int von, int bis, int randzahl[]) {
  int zahl[MAX_TRIALS], zufall, zahler;

    for (zahler = von; zahler < bis; zahler++) {
      zahl[zahler] = zahler;
      randzahl[zahler] = -1;
    }
    for (zahler = von; zahler < bis; zahler++) {
      while (randzahl[zahler] == -1) {
      zufall = getrandom(von, bis-1);
      randzahl[zahler] = zahl[zufall];
      }
      zahl[zufall] = -1;
    }
    return(*randzahl);
}

int set_f_oder_t(int f[], int maxtrials) { // so genau 50% f
  int count, platz;

  for (count = 0; count < maxtrials; count++) {
    f[count] = 0;                                  // setzt alle auf 0
  }
  for (count = 0; count < (maxtrials / 2); count++) {
    do {
      platz = getrandom(0, maxtrials-1); // setzt die Hälfte auf 1
    } while (f[platz] == 1);
    f[platz] = 1;
  }
  return(*f);
}

void set_letters(FILE *datei, int lanz, int f) {
    int  nr, fpos, lpos[16],                    //letterposition
       posnr, alpha = ALPHABET;
    char letter;

    *lpos = set_letter_positions(lpos, lanz);
```

```
    fpos = lpos[0];                                  // an welcher Stelle
f
    for (nr = 0; nr < 16; nr++) {
      letter = ' ';
      if (fpos == nr) {
      if (f == 1) letter = F;    // f einbauen
      else letter = T;           // t einbauen
      }
      else {
      posnr = -1;
      do {
        posnr++;
        if (nr == lpos[posnr])
          do {
            letter = getrandom(0, 26-1) + alpha;
          } while ( (letter == F) || (letter == T) ); //darf kein f/t
sein
      } while ( (nr != lpos[posnr]) && (posnr < lanz-1) ); //bricht
ab,
      }                                    //wenn eine Position
übereinstimmt
    putc(letter, stdout);                  //oder letter_anzahl erreicht
ist
    fputc(letter, datei);
    }
}

int set_letter_positions(int pos[], int lanz) {
  int Posnr, Posnr2, gleichheit;

  for (Posnr = 0; Posnr < lanz; Posnr++) {
      do {
      gleichheit = 0;
      pos[Posnr] = getrandom(0, 16-1);
      Posnr2 = 0;
      while ( (Posnr2 < Posnr) && (gleichheit != 1) ){
        if (pos[Posnr] == pos[Posnr2]) gleichheit = 1;
        else gleichheit = 0;
        Posnr2++;
      }
      } while (gleichheit == 1);
    }
  return(*pos);
}

void syntax() {
  printf("\n\rTRIALS erstellt die Zeichenketten für das SAT -
Programm\n\r");
  printf("Version 2.0, (C) März 1997, Sven Dannemann\n\n\r");
  printf("  Syntax: trials <datei.dat> <Trials im 1. Block>
<Buchstaben pro Trial> ..\n\r");
  printf("Beispiel: trials trials.dat 12 4\n\r");
  printf("    oder: trials trials.dat 10 6 8 12\n\r");
}
```

8.4 TRIALS.DAT

Es folgt der Ausdruck der Textdatei,

die durch das TRIALS-Programm erstellte Zeichenketten enthält.

Die Zeichenketten dienen als Trials im SAT-Programm.

```
50 1 50 3 50 5 50
10
10
  WX GCJ TW LGR
5
 VJ   T S     J
5
     LW  TH   Y
10
YTD H   ACWY HS
1
  T
5
T   Z   LJ     H
10
WK JF S MPJH  A
1
    T
5
 F U    L D R
3
    FC C
3
   V      V  F
10
 TR C YG UR Q DR
10
C  WQX   KFZMA U
5
 G D    T    NM
5
A    M H T  U
1
         T
10
 K KLDACLL   EF
1
            F
10
G IT N   RN NZWA
1
            T
5
  R    F  WU   D
5
 EW   J  F   B
1
          T
10
NAU  MNTDD    XD
5
D   C  C  F   K
10
NG  VV P  J GOJF
3
T  M          K
1
     T
10
DZW   RXBZTM  Q
1
 F
5
   W       H FXG
3
    F  G     R
10
B S FQCIH U  BS
1
 F
1
  F
3
 FS            Q
1
  F
10
 N  PCUZFS  R UP
1
T
10
 D  JPW P  XBLKT
10
NL  MB TU P MK L
3
   F   M     P
3
  D     S    T
3
T      P     E
5
 OR I  FS
3
        F   CE
1
F
5
  O  N   WA F
5
   TB     IB A
3
 F      P   Q
5
S ST QQ
3
  T    X  D
1
       T
5
TX   M R   J
1
             F
5
  KF  Y     BI
1
F
5
 T    LQ  A  D
3
      K     T  K
1
       F
1
         T
5
  BO TCL
10
 DXR OPBYJ  Z  F
10
ZDZDB U   TBXU
5
 RS T  W       S
1
              F
1
   T
10
QPN  F    CKWVNH
5
      D THVY
3
 I  S   F
5
     M   U  F UD
5
N    H X    A F
10
 U UJ KOLX JOT
3
V      E      T
10
S W YX F L I JJH
10
W TG RG O N  UAD
3
  FY   G
3
   J  T     D
1
```

```
         T
10
RK F EUG  NK  NA
10
WOOX V J  JFM  P
1
              T
3
 FH            K
5
M T    L EB
1
              F
1
  T
3
T     Z        Q
3
  N   W  T
1
  T
3
     B    F  H
5
    J    F  BWY
5
  O   Q  N   M T
3
    F    A  P
3
K    N     F
5
  A    R   T ZR
1
          F
3
F         WS
1
        T
1
       T
10
A  YTKQR XY C R
10
KALXATPE  M  V
1
               T
1
    F
5
 Z K    U  F A
10
Z CV B N  CT VXS
10
  CX A EAFU QG N
5
 T  Z Y      I A
10
 S YZUV LDT   VY
5
   F      M ZVR
1
             T
10
  AM   S NSFNQIV
5
T    WS   RQ
3
      N   ET
5
 F  J  AZ H
5
  F P   I X X
3
   Q       F Y
1
              T
3
W  F A
10
IB E XX TY  VSD
3
   F    H     C
5
J   S FY       B
3
      Z T    M
5
  NHOF          D
5
Z W  F       O K
3
 P    F H
10
HKCMO IS R   FQ
3
T  M V
5
 S  TL Z   M
5
        RF Y K X
10
BFG HUO   XRA J
3
     E T       C
10
N Q  WF C UQDSC
3
   Q    F   E
1
      T
3
       U   I  T
3
    C     ST
1
          F
10
VF  CS  WJE BKR
3
 T     G       G
10
HFYXW  J  SJAV
10
 Y HE  PTWZO L H
10
   I  T IZWUDWDZ
3
       B  QT
1
             F
10
E SBJ  L KY EF R
5
 X HU QT
1
  F
3
      B     MF
10
U M TZ  WY UCP A
10
E  FCBBCJ P  D Y
1
           T
3
IT         L
5
L    F   R   D K
3
     Q   U    F
3
   E Y T
1
       F
10
NAYYRB     A QMF
3
   Z   X      F
10
EWQ VQRF C  D L
5
GY   M       FK
5
          MTN  ZQ
1
           T
10
NJTR DOY J  WP
10
FMS VBM VX  R L
1
     T
5
F KJ       I G
1
F
5
  E    Q V R   T
3
    U    F    I
10
HRBLI  R  I  ZTX
5
      FQW   E  A
1
        F
10
H  TXK  M NXMSY
```

```
1
  F
5
   NO  L SF
5
Q  J RR  T
3
         THM
10
 VPZ H  TYUJAB
3
    WF  B
1
            T
3
E     M  F
10
 BQEX  KRQ O FI
```

```
3
  T    A   P
1
                F
1
    F
3
  Q O  T
3
   S E         T
1
              F
5
  S    V  L TO
1
             T
1
   F
```

```
10
Z   V  EWTXO AKU
3
  ET            D
3
      T        ZL
5
  N  F  EV    D
5
  L    PTJ M
10
   V PETKK QMC U
5
 UE U  T   V
10
A  I GFAYR UEI
1
             F
```

8.5 EAMARKEX.C

Es folgt der Quellcode des Eamarkex-Programms,

das aus den EEG-Rohdaten Epochen von je einer Sekunde herausschreibt

und solche, die in ihrer Amplitude in einem oder mehreren Aufnahmekanälen

(die bipolaren Augenkanäle eingeschlossen) 98 µV überschreiten, verwirft.

Das Programm wurde von W. Strik und S. Dannemann erstellt.

```
/*
eamarkex.c  writes marked EEG-epochs to ASCII

cl /AH /O2 /Zp1 /Gt8 /c eamarkex.c
link eamarkex easys;
*/
#include <stdlib.h>
#include <stdio.h>
#include <io.h>
#include <errno.h>
#include <malloc.h>
#include <string.h>
#include <math.h>
#include "ea.h"
#include "data.h"

#define MAX_SEQ 16
#define DEF_WIN 98                  //default artifact window
#define DEF_FRAMES 256

void        end2end_average(float _huge*, int, int);
char        *write_asc(char*,int,int);
int          write_art(char*,int,int);
void        make_tt(char*);
void        get_sequence(char*);
int         artifact(void);
char        *make_fname(char*,char*,int,int);
void        get_args(int,char**);
void        get_targets(void);
FILE        *f_open(char*,char*,long*);
void        errex(char*,char*);

char  msg[]="\n\n"
                                    "eamarkex <infile> [-m<n[-n-*n-
n..>] [-f<n>] [-o<n>] [-a<n>] [-w<n>[#<n>]]\n"
                                    "                  [-b<n-n>] [-
p<n>] [-t] [-e]\n"
                                    "\n"
                                    "vs 1.0, psych neurophysiol
wuerzburg, 26sep97\n"
                                    "-extracts EEG-epochs from binary
Easys d-files\n"
                                    "-epochs with artifacts are marked
with @ in file extension\n"
                                    "\n"
                                    "m marker value/sequence; *=target
(default=first; ?=wildcard)\n"
                                    "f frames to be written to ASCII
(default: 256)\n"
```

```
                                    "o offset from marker
(default: 0)\n"
                                    "a artifact channels (including
event channel!)     (default: 0)\n"
                                    "b bipolar channel (calculated as
n-n =>*.art-file) (default: no bipolar chs)\n"
                                    "w artifact window,
(default: ñ98 uV)\n"
                                    "  #<n> indicates #of frames to be
tested           (default: all frames)\n"
                                    "p position of marker code in out-
file-name          (default: no coding)\n"
                                    "t write time table, suppress out-
put of EEG-data    (default: no time table)\n"
                                    "d end to end average, channelwise
(default: no end to end)\n"
                                    "e write event channel to channel 0
of ASCII outfile;\n"
                                    "";

int fr=DEF_FRAMES,al=-1,art=0,mcp=-1,tt=0,ec=0,tn=0,etoe=0,
            bp[MAX_SEQ], bpn=0,
            sq[MAX_SEQ], sqn=0,
            sqp[MAX_SEQ],sqpn=0;

long fl=0,ofs=0,bpf=0,n=0;
float win=DEF_WIN;
float _huge *ep=NULL;

FILE  *ea;

struct h_info h;
struct m_info *m=NULL,*t=NULL,msq[MAX_SEQ];

void main(int argc, char *argv[])
      {
      char *a;
      int i=0,j=0,fh;

      if(argc<2) errex(msg,"");
      ea= f_open(argv[1],"rb",&fl);
      h= ReadEasysHeader(ea);

      printf("\nreading %s..",argv[1]);

      get_args(argc,argv);

      bpf=(long)h.t*h.nch;
      n=    (long)fr *h.nch;
      ofs*= bpf;
      al= al<0?fr:al;

      if(!(ep= (float*)                         halloc((size_t)n
+1,sizeof(float                       )))) errex("out of memory","");
      if(!(m = (struct m_info*) calloc((size_t)h.nevn+1,sizeof(struct
m_info)))) errex("out of memory","");
      if(!(t = (struct m_info*) calloc((size_t)h.nevn+1,sizeof(struct
m_info)))) errex("out of memory","");

      m= GetEasysMarkers(ea);

      if(tt) make_tt(argv[1]);
                              // make time-table and exit
```

```
      get_targets();

      printf(" %li events, %i targets(",h.nevn,tn);
      for(i=0;i<sqpn;i++) printf("%i%s",sq[sqp[i]],i<sqpn-1?",":"");
      printf(")\n");

      for(i=0;i<tn;i++) {
            if(kbhit()) if(getch()==27) errex(" ESC","");
            printf("\n%s f#%6li m#%3i",argv[1],t[i].fp,t[i].mv);
            fseek(ea,t[i].fp*bpf+h.dpos+ofs,SEEK_SET);
            if(ReadEasys(ea,ep,n)==EOF) errex(" EOF","");
            if(etoe) end2end_average(ep,h.nch,fr);
            printf("=>%s",write_asc(argv[1],i,t[i].mv));
            }
      errex("","");
      }

void end2end_average(float _huge* ep, int nch, int n)
            {
            int i, j;
            double m;

            for(i=0;i<nch;i++) {
                  m=0;
                  for(j=0;j<n;j++) m+=(double)ep[j*nch+i];
                  m/=(double)n;
                  for(j=0;j<n;j++) ep[j*nch+i]-=(float)m;
                  }
            }

void get_args(int c, char **v)
            {
            int i;
            for(i=2;i<c;i++) {
                  switch(v[i][1]){
                        case 'm': get_sequence(v[i]);

      break;
                        case 'f': sscanf(v[i]+2,"%i",&fr);
                                                break;
                        case 'o': sscanf(v[i]+2,"%li",&ofs);
                                                break;
                        case 'a': sscanf(v[i]+2,"%i",&art);
                                                break;
                        case 'b':
sscanf(v[i]+2,"%i%*c%i",&bp[bpn++],&bp[bpn++]); break;
                        case 'w': sscanf(v[i]+2,"%f%*c%i",&win,&al);
                                    break;
                        case 'p': sscanf(v[i]+2,"%i",&mcp);
                                                break;
                        case 't': tt=1;

                  break;
                        case 'e': ec=1;

                  break;
                        case 'd': etoe=1; /*(wie Durchschnitt)*/
                                          break;
                        }
                  }
            }

char *write_asc(char *fn,int run,int mk)
```

```
            {
            int af=0;
            long i,j,f;
            FILE *fp;
            char *fname;

            if(bpn) af= write_art(fn,run,mk);
            if(artifact() || af) fname= make_fname(fn,".as@",run,mk);
            else                                                fname=
make_fname(fn,".asc",run,mk);

            fp= f_open(fname,"w",NULL);
            for(f=0;f<fr;f++) {
                  j=f*h.nch;
                  if(ec) fprintf(fp,"%3.2f\t",ep[j+h.nch-1]);
                  for(i=0;i<h.nch-art;i++)
                        fprintf(fp,"%3.2f%s",ep[j+i],(i+1)%(h.nch-
art)?"\t":"\n");
                  }
            fclose(fp);
            return fname;
            }

int write_art(char *fn,int run, int mk)
            {
            int af=0;
            long i,j,f;
            FILE *fp;
            char *fname;
            float a1,a2;

            for(f=0;f<al;f++) {
                  j=f*h.nch;
                  for(i=0;i<bpn;i+=2) {
                        a1= ep[j+bp[i]]; a2= ep[j+bp[i+1]];
                        if(fabs(a1-a2) > win) af=1;
                        }
                  }
            if(af) fname= make_fname(fn,".ar@",run,mk);
            else   fname= make_fname(fn,".art",run,mk);
            fp= f_open(fname,"w",NULL);
            printf("=>%s",fname);
            for(f=0;f<fr;f++) {
                  j=f*h.nch;
                  for(i=0;i<bpn;i+=2) {
                        a1= ep[j+bp[i]]; a2= ep[j+bp[i+1]];
                        fprintf(fp,"%3.2f%s",a1-a2,i==(bpn-
2)?"\n":"\t");
                        }
                  }
            fclose(fp);
            return af;
            }

char *make_fname(char *fn,char *ext,int run, int mk)
            {
            int i;
            char *fname="            ";

            _splitpath(fn,NULL,NULL,fname,NULL);

            for(i=strlen(fname);i<8;i++) fname[i]='0';
            strcpy(fname+8,ext);
```

```
            if(!(++run)) return fname;
            if(mcp>-1 && mk>=0) fname[mcp]= mk<10?mk+48:mk+55;
            fname[6]= run<100?run/10+48:run/10+55;
            fname[7]= run%10+48;
            return fname;
            }

int artifact(void)
            {
            long i;

            for(i=0;i<al*h.nch;i++) if(fabs(ep[i]) > win && i%h.nch)
return 1;
            return 0;
            }

void get_sequence(char *v)
            {
            int  i=1,j=0;

            while(v[i++]) {
                  if(v[i] == '*') {
                        i++;
                        sqp[j++]=sqn; sqpn++;
                        }
                  if(sq[sqn]= atoi(v+i)) sqn++;
                  //else if(v[i]=='?') sq[sqn++]=0;
                  while(v[i]!='-' && v[i]) i++;
                  }
            }

void get_targets(void)
            {
            int i,j,k;

            if(!sqpn) sqp[sqpn++]= 0;

            for(i=0;i<h.nevn;i++) {
                  j=0;
                  while(j<sqn && (m[i+j].mv == sq[j] || !sq[j])) j++;
                  if(j==sqn) for(k=0;k<sqpn;k++) t[tn++]= m[i+sqp[k]];
                  }
            }

void make_tt(char *fn)
                  {
                  int i;
                  FILE *tt;
                  char *fname;
                  double tpf;

                  fname= make_fname(fn,".tt",-1,-1);
                  tt= f_open(fname,"w",NULL);
                  printf(" => %s",fname);

                  tpf=(double)1000/h.rate;
                  printf(", marker#                          ");
                  fprintf(tt,"marker#\tframe\tvalue\tt\tdt\n");
                  for(i=0;i<h.nevn;i++) {

      printf("\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b%4i,
val:%3i fr#%6li",i,m[i].mv,m[i].fp);
```

```
      fprintf(tt,"%4i\t%6li\t%3i\t%8.1lf\t%6.1lf\n",i,m[i].fp,m[i].mv,
tpf*(double)m[i].fp,tpf*(double) (m[i].fp- (i? m[i-1].fp:0)));
                        }
                  errex("","");
                  }

FILE *f_open(char *fn,char *mode,long *l)
            {
            int fh;
            FILE *f;

            if(!(f= fopen(fn,mode))) errex("file open error: %s",fn);
            fh= fileno(f);
            *l= filelength(fh);
            return f;
            }

void errex(char *t,char *s)
      {
      printf(t,s);
      if(ep) free(ep);
      if(m)  free(m);
      if(t)  free(t);
      fcloseall();
      exit(0);
      }
```

8.6 EAINTPOL.C

Es folgt der Quellcode des Eaintpol-Programms,

welches das grafikgestützte Editieren von artefakthaltigen EEG-Stücken ermöglicht

und das Interpolieren einzelner Elektroden zulässt.

Die Ausgabe erfolgt in ASCII-Dateien.

Das Programm wurde von Th. Müller und S. Dannemann erstellt.

```
// EAINTPOL
// Program for editing eamarkex-AS@-files and
// interpolating electrodes - output in ASC-files

// sd // tm  12.97

// cl /AH /O2 /Zp1 /Gt8 /c eaintpol.c
// link eaintpol features;

// INCLUDES

#include <malloc.h>
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <fcntl.h>
#include <graph.h>
#include <string.h>
#include <io.h>
#include <sys\types.h>
#include <sys\stat.h>
#include <time.h>
#include "features.h"


// DEFINITIONS

#define C 61
#define F 250
#define AW 98

// VARIABLES

FILE  *ast, *asc;

int   n=0, s, nch=C, nf=F, mxp, myp, stretch;
int   intp[10];
float _huge eeg[C+1][F+1];
float dummy[F+1], f=2, ran, art_win= AW;
char  *msg="usage: EAINTPOL <infile (*.as@) [no extension!]>\ndisplays
AS@-file, interpolates electrodes and writes to ASC-file\n";
char  array[]= {3,5,9,9,9,9,9,5,3,0}; // layout of eeg electrodes
char  *fn, *fnasc;
char
      *schriftart="*.fon",
      *schriftgroesse = "1",
      font[60]="bh";     //"t'roman'h100w100",;
struct videoconfig vc;
struct _fontinfo fi;
```

```
// FUNCTIONS && PROCEDURES (pre-declaration)

int   keyboard(void);
int   test(int);
int   exist_channel(int );
int   free_channel(void);
int   forbidden_channel(int);
char  *ftext(char*,int);
void  errex(char*);
void  read_ast(void);
int   read_ini(char *);
void  init_graph(void);
void  mask(void);
void  print_eeg(void);
void  rctext(int, int, char*);
void  Sleep(clock_t);
void  range(void);
void  write_asc();
void  interpolate(void);
void  looking_for_neighbors(int);
int   art_scan(void);
void  init_font(void);
char* setpointer(size_t laenge);

// MAIN PROGRAM

main(int argc, char *argv[]) {

      if(argc<2) errex(msg);

      fn= setpointer(60);
      fnasc= setpointer(60);

      _clearscreen(_GCLEARSCREEN);

      if(!read_ini(argv[0])) {
            printf("ini-file not found\n");
            printf("default values assumed!\a");
            getch();
      }

      stretch= 460/nch;
      f= C/nch*2;

      init_graph();

      strcpy(fn, argv[1]);
      strcpy(fnasc, argv[1]);
      strcat(fn,".as@");
      strcat(fnasc,".asc");

      read_ast();
      init_font();
      mask();
      print_eeg();

      while(keyboard());

      fclose(ast);
```

```
      _setvideomode(_DEFAULTMODE );

      free(fn);
      free(fnasc);
      hfree(eeg);
}


// FUNCTIONS && PROCEDURES

// reads EAINTPOL.INI

int read_ini(char *fini)
      {
      char line[81]="",cbuf[12];
      FILE *ini;
      int i=0;

      strcpy(fini+strlen(fini)-3,"ini");
      if(!(ini= fopen(fini,"r"))) return 0;
      while(fgets(line,80,ini)) {
            if(!strncmp(line,"Channels",5)) sscanf(line,"%*s%i",&nch);
            if(!strncmp(line,"Frames",5)) sscanf(line,"%*s%i",&nf);
            if(!strncmp(line,"Array",5)) sscanf(line,"%*s%s",array);
            if(!strncmp(line,"Art_win",5))
sscanf(line,"%*s%f",&art_win);
      }
      while(array[i]) array[i++]-=48;
      fclose(ini);
      return 1;
      }


// reads data from as@-file

void read_ast (void) {
      int i,j;

      ran=0;

      if(!(ast= fopen(fn, "rt"))) errex("file open error");
      for(i=1;i<=nf;i++) {
                  for(j=1;j<=nch;j++) {
                        fscanf(ast,"%f",&eeg[j][i]);
                  }
      }
      range();
}

// gets data range all over the channels

void range (void) {
      int i,j;
      float mi= 0, mx= 0;

      for(i=1;i<=nf;i++) {
            for(j=1;j<=nch;j++) {
                  if (eeg[j][i]< mi) mi= eeg[j][i];
                  if (eeg[j][i]> mx) mx= eeg[j][i];
            }
      }
      ran= -mi+ mx;
      ran= ran >225 ? 225 : ran;
```

```
      ran= 150;
}


// prints a message

void errex(char *t) {

      _setvideomode(_DEFAULTMODE );
      printf("\a%s",t);

      fcloseall();
      free(fn);
      free(fnasc);
      hfree(eeg);

      exit(0);
}

// initializes graphic screen

void init_graph(void) {
      _setvideomode(_VRES16COLOR);
      _getvideoconfig(&vc);
      mxp=vc.numxpixels;
      myp=vc.numypixels;
      _setcolor(15);
}

// prints screen mask

void mask(void) {
      int i, ycol2=mxp-_getgtextextent(": (un)mark channels"),
                        ycol1=ycol2-_getgtextextent("down");
      char t[12];

      _setcolor(15); //white
      for(i=1;i<=nch;i++) {
            itoa(i, t, 10);
            if(i%2) rctext(40, 2+stretch*i, t);
      }
      _setcolor(14); //yellow
      rctext(ycol1,fi.ascent,"filename:");
      _setcolor(15);
      rctext(ycol1,2*fi.ascent,fn);
      _setcolor(14);
      rctext(ycol1,4*fi.ascent,"Controls");
      rctext(ycol1,6*fi.ascent,"up");
      rctext(ycol2,6*fi.ascent,": increase");
      rctext(ycol1,7*fi.ascent,"down");
      rctext(ycol2,7*fi.ascent,": decrease");
      rctext(ycol1,8*fi.ascent,"0.9");
      rctext(ycol2,8*fi.ascent,": (un)mark channels");
      rctext(ycol2,9*fi.ascent,"  for interpolation");
      rctext(ycol1,10*fi.ascent,"i");
      rctext(ycol2,10*fi.ascent,": interpolate");
      _setcolor(4);
      rctext(ycol1,11*fi.ascent,"Once interpolated");
      rctext(ycol1,12*fi.ascent,"   you must exit!");
      _setcolor(14);
      rctext(ycol1,14*fi.ascent,"Esc");
      rctext(ycol2,14*fi.ascent,": quit");
      _setcolor(9);
```

```
      rctext(mxp-_getgtextextent("  EAINTPOL v1 (12.97)"),myp-
5*fi.ascent,"EAINTPOL v1 (12.97)");
      rctext(mxp-_getgtextextent("Clinical Neurophysiology"),myp-
4*fi.ascent,"Clinical Neurophysiology");
      rctext(mxp-_getgtextextent("   Dept. of Psychiatry"),myp-
3*fi.ascent,"Dept. of Psychiatry");
      rctext(mxp-_getgtextextent(" University of W rzburg"),myp-
2*fi.ascent,"University of W rzburg");
      _setcolor(15);
}

// draws eeg data

void print_eeg (void) {
      int j,i,x,y,s;

      _setcolor(0);
      _rectangle(_GFILLINTERIOR,55,0,480,480);
      _setcolor(15);

      for(j=1;j<=nch;j++) {
            x=60; y=(int)(8+stretch*j+ f*(-eeg[j][1]/ran*16));
            for(s=0;s<10;s++) {
                        if(j==intp[s]) { _setcolor(12); break;}
                        else _setcolor(15);
            }
            _moveto(x,y);
            for(i=1;i<=nf;i++) {
                  x=i*1.5+60;
                  y=(int)(8+stretch*j+ f*(-eeg[j][i]/ran*16));
                  _lineto(x,y);
            }
      }
}

// keyboard handle

int keyboard (void) {
      char c=' ';
      char d=' ';
      char t[50];
      int mark=0, val=0, ec=0, fc=0, fbc=0;
      int ycol2=mxp-_getgtextextent(": (un)mark channels"),
                  ycol1=ycol2-_getgtextextent("down");

      long i;

      while ((c=getch()) !=27) {
            switch(c) {

                  // start interpolation

                  case 'i':
                        if (free_channel()==0) {
                              printf("\a");return 1;
                        }
                        interpolate();

                        if (art_scan()) {
                              _setcolor(4);
                              rctext(ycol1,20*fi.ascent,"Despite inter-
polation,");
```

```
                        rctext(ycol1,21*fi.ascent,"values exceed
artefact");
                        itoa((int)art_win,t, 10);
                        rctext(ycol1,22*fi.ascent,"       window
of");
                        rctext(ycol1+23,23*fi.ascent,t);
                        rctext(ycol1,24*fi.ascent,"        mi-
croV!");
                        rctext(ycol1,25*fi.ascent,"Interpolation
forbidden!");
                        printf("\a");
                        getch();
                        return 0;
                  }

                  mask();
                  print_eeg();

                  rctext(ycol1,20*fi.ascent,"Save interpolation
(y/n)?");
                  while ((d=getch()) !=27) {
                        switch(d) {
                              case 'N': case 'n': return 0;
                              default:    write_asc(); return 0;
                        }
                  }
                  return 0;

            // numbers (interpolation channels)

            case 49: case 50: case 51:
            case 52: case 53: case 54: case 55:
            case 56: case 57:


                  val= c -48;
                  Sleep(300);
                  if (kbhit()) val=val*10+(getch()-48);
                  if (val<1 || val> nch) return 1;

                  ec= exist_channel(val);
                  fc= free_channel();
                  fbc= forbidden_channel(val);

                  if (ec>(-1)) {
                        intp[ec]=0;
                        _setcolor(0);
                        rctext(30,stretch*(val+1)-6,"x");
                  }
                  else if (fc>(-1) && fbc==0) {
                        intp[fc]=val;
                        _setcolor(12);
                        rctext(30,stretch*(val+1)-6,"x");
                  }

                  else printf("\a");

                  mask();
                  print_eeg();
                  return 1;

            // arrows
```

```
                  case 0:
                        switch(getch()) {
                              case 72: f= f>19.5 ? (f=20) : (f+=0.5);

                                    mask();
                                    print_eeg();
                                    return 1;

                              case 80: f= f<1 ? (f=0.5) : (f-=0.5);

                                    mask();
                                    print_eeg();
                                    return 1;
                                    default: break;

                        }
                  default: break;
            }
      }
      return 0;
}

// waiting for a while

void Sleep(clock_t wait) {
      clock_t goal;

      goal= wait+ clock();
      while(goal> clock());

}

// looks for marked channels

int exist_channel(int ch) {
      int i;

      if(ch==0) return 0;
      for (i=0;i<10;i++) if (intp[i]==ch) return i;
      return -1;

}

// looks for free channels

int free_channel(void) {
      int i;

      for (i=0;i<10;i++) if (!intp[i]) return i;
      return -1;

}

// looks for other interpolation channels

int forbidden_channel(int cha) {
      float i=0;
      int n=0, channel;
      struct xy chloc={0,0}, rc={0,0};

      chloc= get_coord(array, cha-1);

      rc.y= chloc.y;
```

```
      for(i=-1;i<2;i+=2) {
            rc.x= chloc.x+i;
            channel= get_channel(array, rc);
            if(exist_channel(channel)>0) return 1;
            if(channel>0) n++;
      }
      rc.x= chloc.x;
      for(i=-1;i<2;i+=2) {
            rc.y= chloc.y+i;
            channel= get_channel(array, rc);
            if(exist_channel(channel)>0) return 1;
            if(channel>0) n++;
      }
      if (n<3) return 1;
      return 0;
}


// prints texts on given coordinates (columns, rows)

void rctext (int row, int col, char *text) {
      _moveto(row, col);
      _outgtext(text);
}


// writes data to ASC-file

void write_asc(void) {

      int i,j,v;
      int ycol2=mxp-_getgtextextent(": (un)mark channels"),
                  ycol1=ycol2-_getgtextextent("down");

      FILE *asc;

      if(!(asc= fopen(fnasc, "wt"))) errex(fnasc);
      rctext(ycol1,30*fi.ascent,"writing ASC-file..");


      for(i=1;i<=nf;i++) {
            for(j=1;j<=nch;j++) fprintf(asc,"%4.2f\t",eeg[j][i]);
            fprintf(asc,"\n");
            }
      fclose(asc);
}


int art_scan(void) {
      int i,j;
      float max_val=0;

      for(i=1;i<=nf;i++) {
            for(j=1;j<=nch;j++) {
                  if ((fabs((double)eeg[j][i]))> max_val)
                              max_val= fabs((double)eeg[j][i]);
            }
      }
      if (max_val > art_win) return 1;
      return 0;

}
```

```
void interpolate(void) {

      int i,j;

      for(i=0;i<10;i++) {
            if(intp[i]==0) continue;
            looking_for_neighbors(intp[i]);
            }
      intp[0]= intp[1]= intp[2]= intp[3]= intp[4]= intp[5]=
            intp[6]= intp[7]= intp[8]= intp[9]= 0;
      }


void looking_for_neighbors(int ch) {

      int i,j,lch=0,r=0;
      struct xy chloc={0,0}, rc={0,0};

      for(i=1;i<=nf;i++) dummy[i]=0;
      chloc= get_coord(array, ch-1);

      rc.y= chloc.y;
      for(i=-1;i<2;i+=2) {
            rc.x= chloc.x+i;
            lch= get_channel(array, rc);
            if (lch) {
                  for(j=1;j<=nf;j++) dummy[j]+= eeg[lch][j];
                  r++;
            }
      }
      rc.x= chloc.x;
      for(i=-1;i<2;i+=2) {
            rc.y= chloc.y+i;
            lch= get_channel(array, rc);
            if (lch) {
                  for(j=1;j<=nf;j++) dummy[j]+= eeg[lch][j];
                  r++;
            }
      }
      for(i=1;i<=nf;i++) eeg[ch][i]= dummy[i]/(float)r;

}


void init_font(void) {
      if(_registerfonts(schriftart) < 0 ) errex("no font in working
directory");
      strcat(font, schriftgroesse);
      _setfont(font);
      _getfontinfo(&fi);
}



char* setpointer(size_t laenge) {
  char *pointer;

  if ((pointer = (char *) malloc(laenge)) == NULL)
                  errex("exceeds memory\n");
  return(pointer);
}
```

8.7 AVERAGE.C

Es folgt der Quellcode des Average-Programms,

das die Durchschnittswerte von EEG-Dateien berechnet.

Das Programm wurde von W. Strik und S. Dannemann erstellt.

```
// AVERAGE.c

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <malloc.h>
#include <limits.h>
#include <plib.h>

#define MAX_LINE SHRT_MAX-1

void get_args(int,char**);
long count_items(FILE*);
void errex(char*,char*);
void test_format(FILE*);
long count_vars(FILE*);
long count_recs(FILE*);

double _huge *ep=NULL;
int header=0;
char hline[MAX_LINE], line[MAX_LINE];

char *msg= "average <file-list> <outfile>\n"
                                "\n"
                                "calculates means of files listed in
<file-list>\n"
                                "all files must have the same format!\n"
                                "\n"
                                "writes header of first infile (if pre-
sent) to outfile\n"
                                "\n"
                                "\n"
                                "(c) 13feb98 psychiat neurophys univ
wuerzburg\n ";

long v=0,c=0,vc=0;


main(int argc, char *argv[])
            {
            char inf[_MAX_PATH];
            long i=0,n=0;
            float val;
            FILE *fl,*in,*out;

            get_args(argc,argv);

            if(!(fl= fopen(argv[1], "r" ))) errex ("\n%s not
found",argv[1]);
            if(!(out=fopen(argv[2], "w" ))) errex ("\ncan not open
%s",argv[2]);
```

```
            test_format(fl);
            if(header) fprintf(out,"%s",hline);

            while(fscanf(fl,"%s",inf)!=EOF) {
                  i=0;
                  if(!(in= fopen(inf, "r"))) errex ("%s not
found!",inf);
                  printf("\n#%2li:%12s..",n+1,inf);
                  if(header) fgets(hline,MAX_LINE,in);
                  while(fscanf(in,"%f",&val) != EOF && i<vc) ep[i++]+=
(double) val;
                  while(fscanf(in,"%f",&val) != EOF) i++;
                  if(i!=vc) errex(" => data error: %s val-
ues",ltoa(i,line,10));
                  fclose(in);
                  n++;
                  }
            fclose(fl);
            printf("\n\n=>writing %s..\n",argv[2]);
            for(i=0;i<vc;i++) fprintf(out,"%8.4f%s",ep[i]/ (double)
n,(i+1)%v?"\t":"\n");
            fclose(out);
            errex("","");
            }


void test_format(FILE *fl)
            {
            char fn[_MAX_PATH];
            FILE *f;
            float val;
            long i=0;

            fscanf(fl,"%s",fn);
            if(!(f= fopen(fn,"r"))) errex("cannot open %s",fn);
            printf("\nchecking %s: ",fn);

            v= count_vars(f);
            c= count_recs(f);
            vc=v*c;

            printf("%s; %li x %li = %li values; %li bytes
mem\n",header?"header found":"no header",v,c,vc,vc*sizeof(double));
            if(!(ep= (double _huge*)halloc(vc+1,sizeof(double _huge))))
errex ("\nout of memory","");

            fclose(f);
            rewind(fl);
            }

long count_vars(FILE *f)
            {
            char *c;
            long i=0;

            fgets(hline,MAX_LINE,f);
            if(!num(*snextf(hline))) {
                  header=1;
                  fgets(line,MAX_LINE,f);
                  }
            else strcpy(line,hline);
            srewind();
```

```
            while(sscanf(snextf(line),"%*f")!=EOF) i++;

            rewind(f);
            return i;
            }

long count_recs(FILE *f)
            {
            long i=0;

            if(header) fgets(hline,MAX_LINE,f);
            while(fgets(line,MAX_LINE,f)) i++;
            rewind(f);
            return i;
            }


void get_args(int c, char **v)
            {
            int i;

            if(c<3) errex(msg,"");
            for(i=3;i<c;i++) {
                  switch(v[i][1]){
                        default: break;
                        }
                  }
            }

void errex(char *t,char *s)
            {
            printf(t,s);
            fcloseall();
            if(ep) hfree(ep);
            exit(0);
            }
```

8.8 EPEAK.C

Es folgt der Quellcode des Epeak-Programms,

das die Parameter der evozierten Potentiale

am Maximum in vordefinierten Fenstern berechnet.

Das Programm wurde von W. Strik und S. Dannemann erstellt.

```
// Calculates EP parameters at peak in predefined window
// extracts trajectories
/*
cl /AL /c epeak.c
link epeak features;
*/

//ws 1/96, 1/98, 2/98

#include <stdio.h>
#include <stdlib.h>
#include <malloc.h>
#include <math.h>
#include <string.h>
#include <features.h>

#define EXIST(name) !access(name, 0)

#define PI (float) 3.14159265
#define C 21
#define F 256
#define PRESTIM 55                 // 220ms
#define CZ 10
#define PZ 15
#define MINFR 67         // 270ms after stimulus; prestim not consid-
ered.
#define MAXFR 117        // 470ms after stimulus;
#define SAMPR 250        // sampling rate

#define GFP 0
#define AMP 1
#define CZP 2
#define PZP 3
#define MX1 4
#define MY1 5
#define MX2 6
#define MY2 7
#define CX1 8
#define CY1 9
#define CX2 10
#define CY2 11
#define MAP 12
#define FEA 13

#define NOUTF 14
#define NFEA      21

char *xt[]= {".gfp",
                                    ".amp",
                                    ".cz",
                                    ".pz",
```

```
                                        ".mx1",
                                        ".my1",
                                        ".mx2",
                                        ".my2",
                                        ".cx1",
                                        ".cy1",
                                        ".cx2",
                                        ".cy2",
                                        ".map",
                                        ".fea"};

char feahead[]= "\tWindowStart"                    // 19 variables
                                                "\tWindowEnd"
                                                "\tGfpLat"
                                                "\tCzLat"
                                                "\tPzLat"
                                                "\tGfpVal"
                                                "\tAmp"
                                                "\tCzAmp"
                                                "\tPzAmp"
                                                "\tMx1"
                                                "\tMy1"
                                                "\tMx2"
                                                "\tMy2"
                                                "\tCx1"
                                                "\tCy1"
                                                "\tCx2"
                                                "\tCy2"
                                                "\tCx"
                                                "\tCy"
                                                "\tDist"
                                                "\tAngle"
                                                "";

char msg[]=             "
ws,2/98"
                                                "\n\n"
                                                "usage: epeak <ep-file>
<outfile> [-s<n>] [-e<n>] [-p<n>] [-r<n>] [-f<n>]\n"
                                                "
[-w<n,n>] [-a<n1n2n3.nm>\n"
                                                "\n"
                                                "<outfile> no exten-
sion!\n"
                                                "          appends new
data on existing files\n"
                                                "\n"
                                                "s  start-frame for
peak analysis (default:  67 after stimulus)\n"
                                                "e  end-frame
(default: 117)\n"
                                                "p  pre-stimulus inter-
vall        (default:  55)\n"
                                                "r  A/D rate
(default: 250)\n"
                                                "f  number of frames
(default: 256)\n"
                                                "w  electrode numbers
(first=0) for 2 waveforms\n"
                                                "
(default: 10,15 = Cz & Pz in 10/20 system)\n"
                                                "a  array; Nx=number of
electrode colums per row #x, m=number of electrode rows\n"
```

```
                                                    "
(default: 35553 = 10/20, 21 channels)\n"
                                                    "\n\n";


void MakeCurve(float*,int,int,int,int,float*);
void MakeAmp(float*,int,int,float*);
int  get_peak(float*,int,int);
void get_val(void);
void get_args(int,char**);
int  get_mem(void);
void free_mem(void);
void errex(char*,char*);
void open_outfiles(char**);
int  read_ep_file(char*);
void write_out(int);
float pol_angle(float,float,float,float);

char array[]= {3,5,5,5,3,0};
struct descript *ext, *dsc;
float *eeg, *val,sampr=(float)SAMPR,digt=(float)1000/SAMPR;
int minfr=MINFR,maxfr=MAXFR,prestim=PRESTIM,ch=0,fr=F,cz=CZ,pz=PZ;

FILE *outf[NOUTF];

main(int argc, char *argv[])
      {
      int i,j,exist=0;

      if(argc<3) errex(msg,argv[0]);
      get_args(argc,argv);
      if(!get_mem()) errex("out of memory","");

      open_outfiles(argv);
      if(exist= read_ep_file(argv[1])) get_val();
      write_out(exist);
      free_mem();
      }

void MakeAmp(float *eeg, int ch, int fr, float *curve)
            {
            int i;
            float max,min;

            for(i=0;i<ch*fr;i++) {
                  if(!(i%ch)) {
                        max= min= eeg[i];
                        }
                  else {
                        max= eeg[i] > max? eeg[i] : max;
                        min= eeg[i] < min? eeg[i] : min;
                        curve[i/ch]= max-min;
                        }
                  }
            }


void get_val(void)
            {
            float cx1,cy1,cx2,cy2;
            int i, gfpp, czp, pzp;

            printf("writing features..\n");
```

```
            MakeAmp  (eeg, (int)ch, (int)fr,
                                        val+AMP*fr);
            MakeCurve(eeg, (int)ch, (int)fr, (int)cz, (int)prestim,
val+CZP*fr);
            MakeCurve(eeg, (int)ch, (int)fr, (int)pz, (int)prestim,
val+PZP*fr);

            Features(eeg, (int)ch * (int)fr, array, val, NULL, ext,
dsc);

            for(i=0;i<fr;i++) val[MX1*fr+i]= ext[i].b.x;
            for(i=0;i<fr;i++) val[MY1*fr+i]= ext[i].b.y;
            for(i=0;i<fr;i++) val[MX2*fr+i]= ext[i].a.x;
            for(i=0;i<fr;i++) val[MY2*fr+i]= ext[i].a.y;

            for(i=0;i<fr;i++) val[CX1*fr+i]= dsc[i].b.x;
            for(i=0;i<fr;i++) val[CY1*fr+i]= dsc[i].b.y;
            for(i=0;i<fr;i++) val[CX2*fr+i]= dsc[i].a.x;
            for(i=0;i<fr;i++) val[CY2*fr+i]= dsc[i].a.y;

            gfpp= get_peak(val+GFP*fr, minfr+prestim, maxfr+prestim );
            czp=  get_peak(val+CZP*fr, minfr+prestim, maxfr+prestim );
            pzp=  get_peak(val+PZP*fr, minfr+prestim, maxfr+prestim );

            for(i=0;i<ch;i++) val[MAP*fr+i]= eeg[gfpp*ch+i];

            val[FEA*fr+0]= (float)minfr*digt;        // window and la-
tencies
            val[FEA*fr+1]= (float)maxfr*digt;
            val[FEA*fr+2]= (float)(gfpp-prestim)*digt;
            val[FEA*fr+3]= (float)( czp-prestim)*digt;
            val[FEA*fr+4]= (float)( pzp-prestim)*digt;

            val[FEA*fr+ 5]= val[GFP*fr+gfpp];
      // amplitudes
            val[FEA*fr+ 6]= val[AMP*fr+gfpp];
            val[FEA*fr+ 7]= val[CZP*fr+ czp];
            val[FEA*fr+ 8]= val[PZP*fr+ pzp];

            val[FEA*fr+ 9]= val[MX1*fr+gfpp];
      // extrema:  negative
            val[FEA*fr+10]= val[MY1*fr+gfpp];
            val[FEA*fr+11]= val[MX2*fr+gfpp];
      //                             positive
            val[FEA*fr+12]= val[MY2*fr+gfpp];

            val[FEA*fr+13]= cx1= val[CX1*fr+gfpp];
            // centroids: negative
            val[FEA*fr+14]= cy1= val[CY1*fr+gfpp];
            val[FEA*fr+15]= cx2= val[CX2*fr+gfpp];
            //                                  positive
            val[FEA*fr+16]= cy2= val[CY2*fr+gfpp];

            val[FEA*fr+17]= (cx1+cx2)/2;
                  // center of gravity
            val[FEA*fr+18]= (cy1+cy2)/2;

            val[FEA*fr+19]= (float)sqrt((cy2-cy1)*(cy2-cy1)+(cx2-
cx1)*(cx2-cx1)); // distance
            val[FEA*fr+20]= pol_angle(cx1,cy1,cx2,cy2);
                   // angle
            }
```

```
void MakeCurve(float *eeg, int ch, int fr, int el, int prestim, float
*curve)
            {
            int i;
            double bl=0;

            for(i=0;i<prestim;i++) bl+= (double) eeg[i*ch+el];
            if(prestim) bl/=(double)prestim;
            for(i=0;i<fr;i++) curve[i]= eeg[i*ch+el]- (float) bl;
            }


int get_peak(float *val,int minfr,int maxfr)
            {
            int i, frame;
            float max;

            max= val[(frame=minfr-1)];
            for(i=minfr;i<maxfr;i++) if(val[i]>max) max=val[(frame=i)];
            return frame;
            }


void get_args(int c, char **v)
            {
            int i,j=0;

            for(i=3;i<c;i++) {
                  switch(v[i][1]){
                        case 's': sscanf(v[i]+2,"%i",&minfr);
                                          break;
                        case 'e': sscanf(v[i]+2,"%i",&maxfr);
                                          break;
                        case 'p': sscanf(v[i]+2,"%i",&prestim);
                                    break;
                        case 'r': sscanf(v[i]+2,"%f",&sampr);
digt=1000/sampr;  break;
                        case 'f': sscanf(v[i]+2,"%i",&fr);
                                                break;
                        case 'w': sscanf(v[i]+2,"%i%*c%i",&cz,&pz);
                              break;
                        case 'a': sscanf(v[i]+2,"%s",array);
                                                      while(array[j])
array[j++]-=48;                                       break;
                        }
                  }
            j=0;
            while(array[j]) ch+= array[j++];
            }

int get_mem(void)
            {
            if(!(eeg=(float _huge*)       halloc((long)    (ch*fr+1)
      ,sizeof(float))))
      {printf("1");return 0;}
            if(!(val=(float*)                           cal-
loc((size_t)(NOUTF*fr+1),sizeof(float))))
      {printf("2");return 0;}
            if(!(ext=(struct descript*)calloc((size_t) fr+1
      ,sizeof(struct descript)))) {printf("3");return 0;}
            if(!(dsc=(struct descript*)calloc((size_t) fr+1
      ,sizeof(struct descript)))) {printf("4");return 0;}
```

```
            return 1;
            }

void free_mem(void)
            {
            free(eeg);
            free(val);
            free(ext);
            free(dsc);
            }


void open_outfiles(char **v)
      {
      char cbuf[60];
      int i,j,exist=0;

      for(j=0;j<NOUTF;j++) {
            strcpy(cbuf,v[2]); strcat(cbuf,xt[j]);
            if(EXIST(cbuf))  exist=1;
            if(!(outf[j]= fopen(cbuf, "a"))) errex("file open error
(%s)",cbuf);
            if(!exist) {
                  switch(j) {
                        case MAP: for(i=0;i<ch;i++)
fprintf(outf[MAP],"\t%i",i+1); break;
                        case FEA: fprintf(outf[FEA],"%s",feahead);
                                          break;
                        default : for(i= 0;i<fr-1;i++)
fprintf(outf[j],"\t%4.1f",(float)(i-prestim)*digt); break;
                        }
                  }
            fprintf(outf[j],"\n%s",v[1]);
            }
      }

int read_ep_file(char *fn)
            {
            int i=0;
            FILE *f;

            printf("reading %s.. ",fn);

            if(!(f= fopen(fn, "r"))) { printf("not found!\n"); return
0; }

            while(i<fr*ch && fscanf(f,"%f",eeg+i)!=EOF) i++;
            if(i< ch*fr || fscanf(f,"%f",NULL)!=EOF) errex("wrong # of
data points!","");
            fclose(f);

            return 1;
            }

void write_out(int flag)
      {
      int i,j;

      for(j=0;j<NOUTF;j++) {
            if(!flag) fprintf(outf[j],"\tnot found");
            else
                  switch(j) {
```

```
                        case MAP: for(i=0;i<ch  ;i++)
fprintf(outf[j],"\t%3.2f",val[j*fr+i]); break;
                        case FEA: for(i=0;i<NFEA;i++)
fprintf(outf[j],"\t%6.4f",val[j*fr+i]); break;
                        default : for(i=0;i<fr-1;i++)
fprintf(outf[j],"\t%3.2f",val[j*fr+i]); break;
                        }
            fclose(outf[j]);
            }
      }


void errex(char *t,char *s) {
      printf(t,s);
      free_mem();
      exit(0);
      }


float pol_angle(float ax,float ay,float bx,float by)
      {
      double d1, d2;

      d1=(double)(ax-bx);  d2= (double)(ay-by);
      return(d2? 90- (float)(360* atan(d1/d2)/(2*PI)) : 0);
      }
```

Danksagung

Allen, die mich bei der Anfertigung der vorliegenden Dissertation unterstützt haben, danke ich sehr herzlich, ganz besonders meinen Eltern Dagmar und Rolf.

Mein Dank gebührt auch Herrn Prof. Werner Strik und Herrn Dr. Thomas Müller, die mich angeleitet haben und immer wieder für Fragen freundlich zur Verfügung standen.